Large-scale Machine Learning in High-dimensional Datasets

Hansen, Toke Jansen

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Large-scale Machine Learning in High-dimensional Datasets

Toke Jansen Hansen

Kongens Lyngby 2013
PhD-2013-300

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Building 303B, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253031
compute@compute.dtu.dk
www.compute.dtu.dk PhD-2013-300

Summary (English)

Over the last few decades computers have gotten to play an essential role in our daily life, and data is now being collected in various domains at a faster pace than ever before. This dissertation presents research advances in four machine learning fields that all relate to the challenges imposed by the analysis of big data.

In the field of *kernel methods*, we present an information-based denoising technique based on semi-supervised kernel Principal Component Analysis (PCA), that incorporates label information into the kernel PCA objective. Effectively, this guides the low-rank representation towards relevant components, while exploiting intrinsic manifold structures exposed by the data. In the same field, we also introduce a scalable randomized heuristic for optimizing kernel hyperparameters, that is based on maximizing the Minimum Enclosing Ball (MEB) of the class means in the associated Reproducing Kernel Hilbert Space (RKHS).

In the field of *spectral methods*, we introduce semi-supervised eigenvectors of a graph Laplacian, that inherit many of the properties that characterize the global eigenvectors, but by using side-information in the form of a seed set, the semi-supervised eigenvectors are better at modeling local heterogeneities.

In the field of *machine learning for neuroimaging*, we introduce learning protocols for real-time functional Magnetic Resonance Imaging (fMRI) that allow for dynamic intervention in the human decision process. Specifically, the model exploits the structure of fMRI data by incorporating a temporal Gaussian Process (GP) smoothness prior, which reduces model degeneracy caused by mislabeled data samples.

Finally, in the field of *topic modeling*, we introduce a Graphics Processing Unit (GPU) accelerated framework for co-clustering in large-scale sparse bipartite networks. By implementing the Infinite Relational Model (IRM) in this framework we achieve speedups of two orders of magnitude compared to estimation based on conventional processors

Summary (Danish)

I løbet af de seneste årtier er computere kommet til at spille en væsentlig rolle i vores daglige liv, og data bliver nu indsamlet i forskellige domæner i et hurtigere tempo end nogensinde før. Denne afhandling præsenterer forskningsresultater i fire overordnede maskinlæringsfelter, som alle vedrører de udfordringer der ligger i analysen af store datasæt.

Inden for *kernelmetoder* præsenterer vi en informationsbaseret støjreduktionsteknik baseret på semi-supervised kernel principal komponentanalyse [Principal Component Analysis (PCA)], som inkorporerer sideinformationer omkring data. Effektivt guider dette lavrangsrepræsentationen mod komponenter, der er relevante, og samtidig udnyttes eksponerede glatte strukturer i data. Inden for samme felt præsenterer vi en skalerbar randomiseret heuristisk til at optimere kernel hyperparametre, der er baseret på maksimering af den mindste omsluttende kugle [Minimum Enclosing Ball (MEB)] udspændt af gennemsnittet for individuelle klassestrukturer, hvor de underlæggende data er repræsenteret i et reproducerende kernel Hilbert rum [Reproducing Kernel Hilbert Space (RKHS)].

Inden for *spektrale metoder* introducerer vi semi-supervised egenvektorer for en graf Laplacian, der arver mange af de egenskaber, der kendetegner de globale egenvektorer, men ved at inkorporere sideinformationer i optimeringsproblemet, er semi-supervised egenvektorer bedre til at modelere lokale heterogeniteter.

Inden for *maskinlæring i neuroimaging* introducerer vi maskinlæringsprotokoller for realtids funktionel magnetisk resonans [functional Magnetic Resonance Imaging (fMRI)], der giver mulighed for dynamiske indgreb i den menneskelige beslutningsproces. Specifikt udnytter modellen strukturen af fMRI data ved at inkor-

porere en tidslig Gausisk process [Gaussian Process (GP)] glatheds-prior, hvilket reducerer modeldegenerering forårsaget af upræcise sideinformationer.

Slutteligt, inden for *emnemodellering*, introducerer vi et grafikkortaccelereret [Graphics Processing Unit (GPU)] system for detektion af klyngedannelser i todelte storskala netværk med sparsomme observationer. Ved at implementere den uendelige relationelle model [Infinite Relational Model (IRM)] i dette system opnår vi hastighedsforbedringer i to størrelsesordener sammenlignet med modelestimering baseret på konventionelle processorer.

List of Publications

Contributions included in this thesis

[A] Toke Jansen Hansen, Morten Mørup, and Lars Kai Hansen. Non-parametric Co-clustering of Large Scale Sparse Bipartite Networks on the GPU. *Proceedings of 2011 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2011.
doi: 10.1109/MLSP.2011.6064611.

[B] Toke Jansen Hansen, Trine Julie Abrahamsen, and Lars Kai Hansen. A randomized heuristic for kernel parameter selection with large-scale multi-class data. *Proceedings of 2011 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2011.
doi: 10.1109/MLSP.2011.6064582.

[C] Toke Jansen Hansen, Lars Kai Hansen, and Kristoffer Hougaard Madsen. Decoding Complex Cognitive States Online by Manifold Regularization in Real-Time fMRI. In *Machine Learning and Interpretation in Neuroimaging (MLINI)*. Springer Berlin Heidelberg, 2012. 76-83.
doi: 10.1007/978-3-642-34713-9_10.

[D] Toke Jansen Hansen, and Michael Mahoney. Semi-supervised Eigenvectors for Locally-biased Learning. *Advances in Neural Information Processing Systems (NIPS)* 25, 2012. 2537-2545.

[E] Toke Jansen Hansen, Trine Julie Abrahamsen, and Lars Kai Hansen. Information-based Kernel PCA Denoising by Semi-supervised Manifold Learning. Submitted to *Pattern Recognition Letters*, 2013.

[F] Toke Jansen Hansen, and Michael Mahoney. Semi-supervised Eigenvectors for Large-scale Locally-biased Learning. Submitted to *Journal of Machine Learning Research (JMLR)*, 2013.

[G] Jens Brehm Nielsen, Bjørn Sand Jensen, and Toke Jansen Hansen. Personalized Audio Systems - a Bayesian Approach. Work in progress; will be submitted to *Audio Engineering Society (AES)*, 2013.

Contributions not included in this thesis

[A] Toke Jansen Hansen, Lars Kai Hansen, and Kristoffer Hougaard Madsen. A Kernel Based Searchlight Heuristic with Real-time Applications. *Abstract/Poster at the 16th Annual Meeting of the Organization for Human Brain Mapping (HBM)*, 2010.

[B] Toke Jansen Hansen, Morten Mørup, and Lars Kai Hansen. Large Scale GPU Based Inference for the Infinite Relational Model. *Neural Information Processing Systems (NIPS) Workshop on Low-rank Methods for Large-scale Machine Learning*, 2010.

[C] Toke Jansen Hansen, Lars Kai Hansen, and Kristoffer Hougaard Madsen. Decoding Complex Cognitive States Online by Manifold Regularization in Real-Time fMRI. *Neural Information Processing Systems (NIPS) Workshop on Machine Learning and Interpretation in Neuroimaging*, 2011.

[D] Toke Jansen Hansen, Lars Kai Hansen, and Kristoffer Hougaard Madsen. Manifold Regularizing for Modeling a Sequence of Neural Events Leading to a Decision. *Abstract/Poster at the 18th Annual Meeting of the Organization for Human Brain Mapping (HBM)*, 2012.

Dedication

To my children; Liva & William

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark (DTU) in partial fulfillment of the requirements for acquiring the Ph.D. degree.

The thesis consists of a summary report that touches upon current and future directions in the field of machine learning, along with a collection of peer-reviewed scientific papers as well as two new contributions, that at the time of writing undergoes review. The work of this Ph.D. was carried out between 2010 and 2013.

Lyngby, 14-April-2013

Toke Jansen Hansen

Acknowledgements

First of all, I would like to thank my supervisor, Professor Lars Kai Hansen, head of the Cognitive Systems section at DTU Compute, Technical University of Denmark, who has been an inspiration and great help during my time in the group. I truly believe that due to his great skills, enthusiasm and guidance, this thesis has been taken to a whole new level.

I acknowledge all members of the Cognitive Systems group for contributing to an inspiring and comfortable atmosphere, and I especially thank Morten Mørup, Mikkel Nørgaard Schmidt, Kasper Winther Jørgensen, Tommy Sonne Alstrøm, Sune Lehmann, Ole Winther and Jan Larsen for the many fruitful theoretical discussions. Moreover, I also like to thank, Kristoffer Hougaard Madsen, researcher at the Danish Research Centre for Magnetic Resonance at Hvidovre hospital, who has been a great mentor throughout my project.

I also thank my office roommates, Bjarne Ørum Wahlgreen, Trine Julie Abrahamsen, and Tue Herlau for all the great times we had. Furthermore, I specially thank my co-authers, Trine Julie Abrahamsen, Kristoffer Hougaard Madsen, Morten Mørup, Bjørn Sand Jensen, Jens Brehm Nielsen, Michael Mahoney and Lars Kai Hansen for their collaboration and interesting discussions.

I would also like to express acknowledgements to my friends who have been very supportive and showed great interest in my work.

Finally, I thank my mom and dad for always making me feel special, and my fiance, Mette Wacher Elnegaard, and our two children, Liva and William, for being loving and caring.

Contents

Acronyms & Abbreviations

ALOI	Amsterdam Library of Object Images 20
API	Application Programming Interfaces 2
ARD	Automatic Relevance Determination 20
BCI	Brain-Computer Interface 42
BLAS	Basic Linear Algebra Subprograms 2
BOLD	Blood Oxygen Level Dependent 42
CPU	Central Processing Unit 2
CRP	Chinese Restaurant Process 54
CUDA	Compute Unified Device Architecture 2
CV	Cross-Validation 5
EEG	ElectroEncephaloGraphy 46
EM	Expectation Maximization 42
EPI	Echo Planar Imaging 42
fMRI	functional Magnetic Resonance Imaging 3
GLM	General Linear Model 42
GMM	Gaussian Mixture Models 28
GP	Gaussian Process 6
GPU	Graphics Processing Units 2
ICA	Independent Component Analysis 46
IRM	Infinite Relational Model 5

KKT	Karush-Kuhn-Tucker 4
KM	K-Means 46
LDA	Latent Dirichlet Allocation 28
LLE	Local Linear Embedding 46
LR	Logistic Regression 18
MAP	Maximum A Posteriori 19
MCMC	Markov Chain Monte Carlo 2
MEB	Minimum Enclosing Ball 3
MKL	Math Kernel Library 58
MP	Multi Processors 57
MSE	Mean Squared Error 6
OpenCL	Open Computing Language 2
PAC	Primary Auditory Cortex 48
PCA	Principal Component Analysis 3
PMC	Primary Motor Cortex 47
PSD	positive semidefinite 29
QP	Quadratic Programming 19
RKHS	Reproducing Kernel Hilbert Space 3
ROI	Region Of Interest 47
SP	Scalar Processors 57
SFM	Softmax Function Model 44
SGT	Spectral Graph Transducer 33
SNR	Signal-to-Noise Ratio 42
SVM	Support Vector Machine 6
WWW	World Wide Web 2

Chapter 1

Introduction

This introductory chapter is meant to serve as a general motivation, setting the stage for the research contributions of this dissertation, as well as providing the reader with an overview of how the remaining chapters are organized.

1.1 Motivation

Data analysis turns out to be a challenging problem when data sets become so large that commonly used processing tools are infeasible. Nevertheless there is a clear trend towards even larger data sets, as explained by for instance the everlasting growth of the World Wide Web (WWW), where search engines provide clients with enormous amounts of indexed data such as videos, images, and text. Also, quality demands in for example image processing, result in digital cameras with ever increasing resolutions, that will set new standards for television sets, as well as video games. The whole industry is like a closed loop where competition in a particular area, triggers new possibilities and/or demands in another, that again will raise the bar and set new standards.

More recently the computer games industry have sparked the research in computer graphics that ultimately led to specialized devices for beautiful graphics rendering, known as Graphics Processing Units (GPU). The massive market for computer games, as well as the inherit competition, made GPUs fairly inexpensive and thereby available for the general consumer. Interestingly, even though GPUs were specialized for graphics rendering, other research fields discovered that these new consumer devices were not only suitable for rendering, but applicable for general purpose operations, such as Basic Linear Algebra Subprograms (BLAS) and Markov Chain Monte Carlo (MCMC) simulations. Being able to fit such operations within GPUs lead to tremendous performance improvements, that for numerous applications outperformed corresponding Central Processing Unit (CPU) implementations by orders of magnitude. Initial implementations were based on standard graphics rendering Application Programming Interfaces (API), such as shaders, that in terms of readability and implementation were suboptimal. Eventually the industry became aware of the new market and began standardizing the programming paradigm, which among others resulted in Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL). By now these paradigms are fairly mature and as a result GPUs are now powering the fastest supercomputer of the world.

In the same way as the computer games industry pushed the development of GPUs at a high pace, the availability of big data is pushing research towards more scalable algorithm for data analysis. However, are more scalable algorithms really a necessity when computers seem to double their performance[1] every 18 months, as predicted by Moores law? – *Most indeed* – The amount of data increases at a higher pace than Moores law predict the transistor count [Villars et al., 2011]. Obviously the curse of dimensionality will remain true, and

[1] Assuming transistor count translates directly to raw performance.

likely to become and even greater limitation than it is today, as the gap between available data and computer performance will continue to expand. This makes new approximation techniques as well as algorithms that can exploit modern architectures of the multicore era important as ever before.

This dissertation presents research advances in four machine learning fields that all relate to the challenges that are imposed by the analysis of big data. Specifically, advances are presented in; kernel methods, spectral methods, machine learning for neuroimaging, and topic modeling.

Kernel methods refer to nonlinear algorithms for machine learning that exploit what is known as the so-called kernel trick. The methodology has become very popular as the trick naturally can extend classical linear algorithms, to allow for nonlinear mappings, given that the data only enters in the form of inner products. In this dissertation, one contribution related to kernel methods is an information-based denoising approach that is based on semi-supervised kernel Principal Component Analysis (PCA) and the inherently ill-posed pre-image problem. The classical denoising technique relies on a low-rank assumption, in that the relevant signal is assumed to be spanned by the leading kernel PCA components, but when the signal of interest is weak compared to other components, the approach is degenerate. The information-based technique incorporates label-information into the kernel PCA objective, guiding the low-rank representation towards components that are relevant. Moreover, we incorporate a graph regularizer into the pre-image problem to benefit from intrinsic manifold structures when only few labeled samples are available. Another contribution related to the field, considers a randomized heuristic for optimizing kernel hyperparameters. Specifically, we maximize a Minimum Enclosing Ball (MEB) of the class means in the associated Reproducing Kernel Hilbert Space (RKHS), which can be computed efficiently by exploiting randomization.

Spectral methods are widely used for approximating graph partitions, that in general is an NP-complete problem. The approximation is usually based on the smallest eigenvectors of a Laplacian matrix for a graph, and by clustering in this low-rank representation good graph partitions can be found. Inherently eigenvectors are global quantities, and a low-rank representation is likely to fail at modeling minor local heterogeneities. In this dissertation, the contribution related to spectral methods is a methodology for constructing, what we call; semi-supervised eigenvectors of a Laplacian matrix. These semi-supervised eigenvectors inherit many of the nice properties that characterize the global eigenvectors, but by using side-information in the form of a seed set, the semi-supervised eigenvectors are better at modeling local heterogeneities.

Neuroimaging by real-time functional Magnetic Resonance Imaging (fMRI) allows for intervention in the human decision process, thereby allowing for more

sophisticated paradigms than in a traditional offline experimental setup. The human decision process is known to be both complex and influenced by many factors on multiple time scales, as reflected by the numerous brain networks and connectivity patterns involved. In this dissertation, we work towards active learning protocols in neuroimaging and we discuss the possibility of combining real-time fMRI and online machine learning which will allow experimental interventions dependent on the cognitive state of the subject. Specifically we propose a semi-supervised modeling approach, that exploits the smoothness of fMRI data, to effectively avoid model degeneracy caused by mislabeled data samples.

Topic modeling by co-clustering is a problem of both theoretical and practical importance, *e.g.*, in market basket analysis and collaborative filtering, as well as in web scale text processing. In this dissertation, we state the co-clustering problem in terms of non-parametric generative models which can address the issue of estimating the number of row and column clusters from a hypothesis space of an infinite number of clusters. To reach large-scale applications of co-clustering we exploit that parameter inference for co-clustering is well suited for parallel computing. The main contribution is a carefully crafted GPU framework for efficient inference on large-scale sparse bipartite networks and this implementation achieves a speedup of two orders of magnitude compared to conventional CPU model estimation.

1.2 Outline and Contributions

In addition to this very general motivation, the rest of this dissertation serves as an introduction to the general fields that have influenced the work of this thesis, as well as providing compressed overviews of selected publications. In detail, the remainder of this dissertation is organized as follows[2]:

Chapter 2 - Constrained Optimization; introduces general concepts, such as Lagrange multipliers and Karush-Kuhn-Tucker (KKT) conditions. These concepts have played an essential role for many of the contributions of this thesis; particular in [Hansen and Mahoney, 2012, 2013; Hansen et al., 2013].

Chapter 3 - Kernel Methods; introduces the kernel trick that allows existing linear models to handle nonlinear data in a non-parametric computationally efficient manner. This chapter also contain discussions related

[2]Papers are ordered according to submission date.

to [Hansen et al., 2011a] that considers efficient optimization of kernel hyperparameters, as well as [Hansen et al., 2013] that exploits labeled data to improve denoising based on the pre-image problem of kernel PCA.

Chapter 4 - Spectral Methods; introduces the general properties of eigenvectors of a graph Laplacian and normalized cuts. Then in detail, the key contributions of [Hansen and Mahoney, 2012] and [Hansen and Mahoney, 2013] are discussed together with examples demonstrating properties of the methodology.

Chapter 5 - Neuroimaging; introduces machine learning challenges encountered in fMRI analysis, as this particular field has motivated much of the research in this thesis. In particular, we present examples related to [Hansen et al., 2012] that considers label uncertainty in real-time fMRI, as well as examples on how semi-supervised eigenvectors [Hansen and Mahoney, 2013] can be used for information-based data-driven feature extraction.

Chapter 6 - Topic Modeling; introduces the challenges of using GPUs for topic modeling on sparse bipartite graphs, as such architecture are much better suited for dense matrix operations. In particular, this chapter introduces the Infinite Relational Model (IRM) together with results based on the large-scale implementation described in [Hansen et al., 2011a].

Chapter 7 - Conclusions; summarizes the key contributions of this thesis.

Paper A - Non-parametric Co-clustering of Large Scale Sparse Bipartite Networks on the GPU, [Hansen et al., 2011b], presents a GPU accelerated implementation of the co-clustering problem formulated in terms of non-parametric generative models. To reach large-scale applications we exploit that parameter inference for co-clustering is well suited for parallel computing. Specifically we devise a generic GPU framework, in which we formulate the IRM and demonstrate efficient inference on large-scale sparse bipartite networks. In both simulations and on real datasets the implementation achieves a speedup of two orders of magnitude compared to estimation based on conventional CPUs.

Paper B - A Randomized Heuristic for Kernel Parameter Selection with Large-scale Multi-class Data, [Hansen et al., 2011a], considers the challenge of estimating hyperparameters in kernel algorithms when regular Cross-Validation (CV) proves infeasible due to the size of the problem. We present a novel heuristic for finding good estimates that is based on maximizing the MEB of the class means in the associated RKHS, that can be computed efficiently by exploiting randomization. Compared to other distance metrics in the RKHS we find that our randomized approach provides better results together with a highly competitive time complexity.

Paper C - Decoding Complex Cognitive States Online by Manifold Regularization in Real-time fMRI, [Hansen et al., 2012], investigates a methodology for avoiding model degeneracy caused by mislabeled data samples. In real-time fMRI this is particularly important as the natural intervention in the human decision process that can be investigated using online learning, allows for more dynamic paradigms that unfortunately makes it difficult to quantify the subjects brain state at a given point in time. The model exploits a temporal Gaussian Process (GP) smoothness prior, and on synthetic data we demonstrate a significant advantage compared to using a Support Vector Machine (SVM), whereas on real fMRI data we observe indications of improved generalizability.

Paper D - Semi-supervised Eigenvectors for Locally-biased Learning, [Hansen and Mahoney, 2012], proposes a new methodology for locally-biased machine learning. In many applications, one has information about a specific target region of a large data set, and one wants to perform common machine learning and data analysis tasks nearby the pre-specified target region. In such situations, regular eigenvector-based methods tend to have serious difficulties, as they are inherently global quantities. The proposed semi-supervised eigenvectors are successively-orthogonalized directions of maximum variance, constrained on being well-correlated with an input seed set of nodes that are assumed to be provided in a semi-supervised manner. They inherit many of the nice properties that characterizes the leading nontrivial global eigenvectors of a graph Laplacian, for example, they capture slowly varying modes in the data, they are efficiently computable, and they can be used for common machine learning tasks such as kernel-based and semi-supervised learning, *etc.* Using several empirical examples we demonstrate how these semi-supervised eigenvectors can be used to perform locally-biased machine learning.

Paper E - Information-based Kernel PCA Denoising by Semi-supervised Manifold Learning, [Hansen et al., 2013], proposes two approaches for exploiting label information to improve the denoising in problems were side-information, in the form of labeled data is available. First, the classical kernel PCA formulation is augmented by a loss term, leading to an iterative algorithm for finding orthonormal components biased by the class labels. Secondly, we devise a fixed-point iteration scheme for solving the pre-image problem for a manifold warped RKHS. We demonstrate the proposed methods on an image classification problem, where it is shown that incorporating label information decreases the sensitivity to the choice of kernel hyperparameter, and improves the denoising performance as measured by the Mean Squared Error (MSE), indicating that incorporating label information indeed results in a more descriptive manifold representation.

Paper F - Semi-supervised Eigenvectors for Large-scale Locally-biased Learning, [Hansen and Mahoney, 2013], elaborates on the methodology introduced in [Hansen and Mahoney, 2012]. Two new variants of the algorithm are introduced that allows for large-scale data analysis. Specifically, we introduce a solution based on low-rank approximations that is applicable for general machine learning tasks, as well as a diffusion based approximation, that is very scalable as the solution can be obtained from local PageRank diffusions on a graph. We provide several empirical examples that demonstrate how these semi-supervised eigenvectors can be used to perform locally-biased learning, and we discuss the relationship between our results and recent machine learning algorithms that use global eigenvectors of the graph Laplacian.

Paper G - Personalized Audio Systems - a Bayesian Approach, [Nielsen et al., 2013] (work in progress), presents a framework for user-guided parameter estimation, based on active learning protocols using GP regression. Specifically, audio systems are typically equipped with many user-adjustable parameters and in order to find the optimal settings, the user must effectively do high-dimensional optimization with respect to subject preferences. We demonstrate the framework in a real interactive loop where twenty-four subjects are given a personalized setting of a five-band equalizer with thousands of possible settings, and we show that the proposed preference methodology is able to find significantly better solutions than through usual random experimentation.

Chapter 2

Constrained Optimization

This chapter introduces the general principles of constrained optimization, that are preliminary for the majority of results presented in this dissertation.

Lagrange multipliers allow elegant solutions to constrained optimization problems, and they play an essential role in the contributions of this dissertation. Here we introduce the concept of Lagrange multipliers by taking a similar approach as used in [Bishop, 2007].

Let $\boldsymbol{x} \in \mathbb{R}^D$, and let $f(\boldsymbol{x})$ be the objective function with constraint equation $g(\boldsymbol{x}) = 0$. The constraint $g(\boldsymbol{x}) = 0$ is a $(D-1)$-dimensional surface in $\mathbb{R}^D$, and the gradient $\nabla g(\boldsymbol{x})$ is orthogonal to the surface, which can be shown using a Taylor expansion around $\boldsymbol{x}$ together with a nearby point, also on the constraint surface [Bishop, 2007].

We now seek a point on the constraint surface, so that $f(\boldsymbol{x})$ is maximized. For the optimal point $\boldsymbol{x}_{opt}$ the gradient $\nabla f(\boldsymbol{x}_{opt})$ must be orthogonal to the constraint surface, since otherwise $\boldsymbol{x}$ could be moved to a location closer to the optimum. Thus, $\nabla g(\boldsymbol{x}_{opt})$ and $\nabla f(\boldsymbol{x}_{opt})$ must be parallel or antiparallel, and

$$\nabla f(\boldsymbol{x}_{opt}) + \lambda \nabla g(\boldsymbol{x}_{opt}) = 0 \tag{2.1}$$

must hold for some $\lambda \in \mathbb{R}$. In the case where $\lambda \neq 0$, λ is called a *Lagrange multiplier*.

We then define the Lagrangian as

$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}) \tag{2.2}$$

so that $\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda) = 0$, is the constrained stationary condition shown in Eq. 2.1, whereas $\frac{\partial}{\partial \lambda} L(\boldsymbol{x}, \lambda) = 0$ leads to the constraint $g(\boldsymbol{x}) = 0$.

To find the maximum of the function $f(\boldsymbol{x})$ subject to the constraint $g(\boldsymbol{x}) = 0$, we therefore calculate the stationary point of $L(\boldsymbol{x}, \lambda)$ with respect to both $\boldsymbol{x}$ and λ, and solve these equations for $\boldsymbol{x}$.

In the case of inequality constraints of the form, $g(\boldsymbol{x}) \geq 0$, two scenarios can occur. If the constrained stationary point lies in the region where $g(\boldsymbol{x}) > 0$, the constraint is *inactive*, and the stationary condition simply becomes $\nabla f(\boldsymbol{x}) = 0$, corresponding to $\lambda = 0$ in Eq. 2.2. In the other case, the constrained stationary point lies on the boundary, corresponding to the equality constraint $g(\boldsymbol{x}) = 0$, and the constraint is now said to be *active*. Since the optimum is not in the subspace spanned by $g(\boldsymbol{x}) > 0$, the sign of λ must be so that $\nabla f(\boldsymbol{x})$ is oriented away from that subspace. Thus, $\nabla f(\boldsymbol{x})$ and $\nabla g(\boldsymbol{x})$ must be antiparallel

$$\nabla f(\boldsymbol{x}) = -\lambda \nabla g(\boldsymbol{x}), \quad \lambda \geq 0. \tag{2.3}$$

In both of the two scenarios we have that $\lambda g(\boldsymbol{x}) = 0$, since in the former inactive case $\lambda = 0$ whereas in the latter active case $g(\boldsymbol{x}) = 0$.

Maximizing $f(\boldsymbol{x})$ subject to $g(\boldsymbol{x}) \geq 0$, is therefore achieved by maximizing the Lagrangian in Eq. 2.2 subject to the constraints

$$g(\boldsymbol{x}) \geq 0 \tag{2.4}$$

$$\lambda \geq 0 \tag{2.5}$$

$$\lambda g(\boldsymbol{x}) = 0 \tag{2.6}$$

which together are known as the KKT conditions.

In the case where we want to minimize a function $f(\boldsymbol{x})$ subject to an inequality constraint $g(\boldsymbol{x}) \geq 0$, then we simply minimize the Lagrangian function $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) - \lambda g(\boldsymbol{x})$ with respect to $\boldsymbol{x}$, again subject to $\lambda \geq 0$.

2.1 Lagrange Duality

The fundamental goal of duality is to cast a certain problem into another formulation that is more convenient, *e.g.*, in terms of computability.

Let us now consider the following general constrained optimization problem

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^D}{\text{minimize}} \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, \quad \forall i \in \{1, \ldots, M\}, \\ & h_j(\boldsymbol{x}) = 0, \quad \forall j \in \{1, \ldots, P\}, \end{aligned} \tag{2.7}$$

where M and P respectively denote the number of inequality and equality constraints.

We now define the Lagrangian, $L_P : \mathbb{R}^D \times \mathbb{R}^M \times \mathbb{R}^P \mapsto \mathbb{R}$, for the optimization problem

$$L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i=1}^{M} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{P} \nu_j h_j(\boldsymbol{x}), \tag{2.8}$$

which we denote as the Lagrange *primal function* of the problem. Note that since $\nabla f(\boldsymbol{x})$ and $\nabla g_i(\boldsymbol{x})$ are antiparallel, this implies that the inequality Lagrange multiplier must be $\lambda_i \geq 0$.

The Lagrange *dual function*, $L_D : \mathbb{R}^M \times \mathbb{R}^P \mapsto \mathbb{R}$, can be defined as the "minimum" of the primal Lagrangian over $\boldsymbol{x}$

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}\in\Omega} \left(f(\boldsymbol{x}) + \sum_{i=1}^{M} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{P} \nu_j h_j(\boldsymbol{x}) \right) \tag{2.9}$$

where "inf" is the pointwise infimum (greatest lower bound), and where the domain Ω is defined as the nonempty intersection of the constraint functions [Hindi, 2006]. Thus, if the primal function is unbounded below in $\boldsymbol{x}$, the dual function takes the value $-\infty$. According to [Boyd and Vandenberghe, 2004] we shall call parameters $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ for which $L_D > -\infty$ dual feasible.

A crucial observation is that the dual is an affine function of $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$, and it will therefore be a concave function, even though the primal is not a convex function [Boyd and Vandenberghe, 2004].

Now, let $p \in \mathbb{R}$ denote the optimal value of $f(\boldsymbol{x})$ in the minimization problem in Eq. 2.7. An important property of the dual function, is that it yields lower bounds for p, because for any feasible point $\hat{\boldsymbol{x}}$, we have

$$\sum_{i=1}^{M} \lambda_i g_i(\hat{\boldsymbol{x}}) + \sum_{j=1}^{P} \nu_j h_j(\hat{\boldsymbol{x}}) \leq 0, \tag{2.10}$$

since $g_i(\hat{\boldsymbol{x}}) \leq 0$, $h_j(\hat{\boldsymbol{x}}) = 0$ and $\lambda_i \geq 0$. Thus, by adding $f(\hat{\boldsymbol{x}})$ on both sides in the above expression, we recover the primal

$$L_P(\hat{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f(\hat{\boldsymbol{x}}). \tag{2.11}$$

It is therefore immediate, that the dual is a lower bound

$$L_D(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \Omega} L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq L_P(\hat{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f(\hat{\boldsymbol{x}}). \tag{2.12}$$

The optimal lower bound, can then be expressed as a new optimization problem, denoted the Lagrange *dual problem*

$$\begin{aligned} \underset{\boldsymbol{\lambda} \in \mathbb{R}^M, \boldsymbol{\nu} \in \mathbb{R}^P}{\text{maximize}} \quad & L_D(\boldsymbol{\lambda}, \boldsymbol{\nu}), \\ \text{s.t.} \quad & \lambda_i \geq 0, \quad \forall i \in \{1, \ldots, M\}. \end{aligned} \tag{2.13}$$

The dual problem is a convex optimization problem, since the objective is to maximize a concave function, with convex constraints.

If we let $d \in \mathbb{R}$ be the dual optimal point, then due to the result in Eq. 2.12, it will always be that $d \leq p$. This releation is called *weak duality*, and the difference $p - d$ is called the *duality gap*. If $d = p$, we say that *strong duality* holds, and the optimal solution to the primal problem can therefore be expressed in terms of the optimal dual variables of the dual problem, which sometimes is easier to solve than the primal problem.

Strong duality does in general not hold, but if $\{f(\boldsymbol{x})\} \cup \{g_i(\boldsymbol{x}) \mid i \in \{1, \ldots, M\}\}$ are convex functions and the equality constraints are linear, strong duality does

usually hold [Hindi, 2006]. To guarantee strong duality, constraint qualification such as Slater's condition can be exploited. The condition states that strong duality holds if the original problem is convex, and there exists a *strictly feasible* point, *i.e.*, there exists a point $\boldsymbol{x}$ such that the inequalities are strictly satisfied

$$g_i(\boldsymbol{x}) < 0, h_j(\boldsymbol{x}) = 0. \tag{2.14}$$

I refer to [Boyd and Vandenberghe, 2004] for a deeper analysis and proof of Slater's theorem.

To conclude this section on Lagrange duality, let us consider a more symmetrical form of the primal and dual, to gain some intuition about the geometrical structure. By definition, the dual optimal point can be written as

$$d^* = \sup_{\boldsymbol{\lambda},\boldsymbol{\nu}} \inf_{\boldsymbol{x}} L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.15}$$

and the primal optimal point can be written as [Boyd and Vandenberghe, 2004]

$$p^* = \inf_{\boldsymbol{x}} \sup_{\boldsymbol{\lambda},\boldsymbol{\nu}} L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.16}$$

where $\boldsymbol{x} \in \Omega$, $\boldsymbol{\lambda} \in \mathbb{R}_+^M$, $\boldsymbol{\nu} \in \mathbb{R}^P$, and where "sup" is the pointwise supremum (least upper bound).

Because strong duality implies that $p^* = d^*$, it also implies that the minimization and maximization can be interchanged, since

$$\sup_{\boldsymbol{\lambda},\boldsymbol{\nu}} \inf_{\boldsymbol{x}} L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}} \sup_{\boldsymbol{\lambda},\boldsymbol{\nu}} L_P(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.17}$$

Thus, the optimal point corresponds to a saddle point, due to the strong *max-min property* [Boyd and Vandenberghe, 2004].

2.2 Optimality Conditions

We shall now assume that the optimization problem in Eq. 2.7 contains differentiable objective and constraint functions.

If the primal problem is non-convex, but there exist optimal points, $\boldsymbol{x}^*$ for the primal, and $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ for the dual, with a zero duality gap, then the KKT conditions

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{M} \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^{P} \nu_i^* \nabla h_i(\boldsymbol{x}^*) = 0 \tag{2.18}$$

$$g_i(\boldsymbol{x}^*) \leq 0, \quad i \in \{1, \ldots, M\} \tag{2.19}$$
$$\lambda_i^* \geq 0, \quad i \in \{1, \ldots, M\} \tag{2.20}$$
$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, \quad i \in \{1, \ldots, M\} \tag{2.21}$$
$$h_j(\boldsymbol{x}^*) = 0, \quad j \in \{1, \ldots, P\} \tag{2.22}$$

are necessary for any pair of primal and dual variables, in order to be optimal [Boyd and Vandenberghe, 2004]. Thus, the KKT conditions do not imply optimality in the non-convex case, but they must be fulfilled at the optimum.

Finally, if the primal problem is convex, and if Slater's condition applies, then the KKT conditions are necessary and sufficient for optimality [Boyd and Vandenberghe, 2004]. Specifically, in [Hansen and Mahoney, 2012, 2013] the KKT conditions of an optimization ansatz were used to provide valuable insights with respect convexity.

Chapter 3

Kernel Methods

This chapter introduces the kernel trick and related applications. In the general field of kernel methods we introduce [Hansen et al., 2011a] related to the optimization of kernel hyperparameters. With respect to kernel PCA and the pre-image problem we give an introduction to [Hansen et al., 2013] that exploits side-information for improving the manifold representation. Finally, we briefly touch upon [Nielsen et al., 2013] (work in progress) that exploits Bayesian non-parametrics for quantifying uncertainty, with respect to modeling human audio preferences.

During the last few years, kernel methods have had an enormous influence on the developments in nonlinear multivariate machine learning. Especially, the kernel trick has allowed existing linear algorithms, to be modified into nonlinear versions with minimal effort. This section will introduce some of the most important results of kernel methods.

A Hilbert space $\mathcal{H}$ is a real or complex inner product space and is in essence an infinite dimensional euclidian space. The space is closed under addition and scalar multiplication, and obeys the commutative, distributive and associative laws, *i.e.*, the inner product $\langle \cdot, \cdot \rangle$ in $\mathcal{H}$ obeys the following conditions:

$$\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle \tag{3.1}$$
$$\langle \alpha x, y \rangle = \alpha \langle x, y \rangle \tag{3.2}$$
$$\langle x, y \rangle = \langle y, x \rangle \tag{3.3}$$
$$\langle x, x \rangle \geq 0 \tag{3.4}$$
$$\langle x, x \rangle = 0 \Rightarrow x = 0 \tag{3.5}$$

Note that we do not use vector notation (boldface) in the definition and conditions for the inner product, since the exact domain has yet to be defined. The norm $\| \cdot \|$ is defined in terms of the inner product via $\|x\| = \langle x, x \rangle^{1/2}$. Finally, the Hilbert space $\mathcal{H}$ is complete if every Cauchy sequence converges with respect to this norm to an element in the space [Scholkopf and Smola, 2001].

A RKHS is a Hilbert space of functions in which pointwise evaluation is a continuous linear functional. Given a kernel, $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, we shall now construct a Hilbert space such that k is a dot product in that space. We then define the Gram matrix as $K_{ij} = k(x_i, x_j)$, for $x_1, x_2, \ldots, x_N$, and we say that the kernel is positive definite, if the Gram matrix is positive definite. Thus, the determinant of $\boldsymbol{K}$ must be nonnegative.

Now assume that k is a real and positive definite kernel, and that $\mathcal{X}$ is a nonempty set. The *reproducing kernel map* is then defined as a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, written as: $\varphi : \mathcal{X} \mapsto (\mathcal{X} \mapsto \mathbb{R})$ (equivalently $\varphi : \mathcal{X} \mapsto \mathcal{H}$), and defined as: $\varphi(x) \mapsto k(\cdot, x)$.

This basically means, that to each point $x \in \mathcal{X}$ (our original space), we associate the function $\varphi(x)$ that assigns the value $k(x', x)$ to $x' \in \mathcal{X}$. The image of φ can be turned into a vector space, using a linear combination of the form

$$f(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i), \tag{3.6}$$

where $\alpha_i \in \mathbb{R}$. This space is the RKHS, and Eq. 3.6 is due to the *representer theorem* [Kimeldorf and Wahba, 1970].

Now, let $g(\cdot) = \sum_{j=1}^{N'} \beta_j k(\cdot, x'_j)$, then the inner product is defined as

$$\langle f, g \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N'} \alpha_i \beta_j k(x_i, x'_j). \tag{3.7}$$

As a final example of this section we verify that this is actually an inner product in the RKHS, and to do so, we must verify that the above definition fulfills the previous stated conditions. Since symmetry and linearity are the easiest to show, we shall here focus on the final property, namely that $\langle f, f \rangle = 0 \Rightarrow f = 0$.

Because the kernel is *representer of evaluation*, then

$$\langle k(\cdot, x), f \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x) = f(x). \tag{3.8}$$

Thus, if we replace f with a kernel in the above expression, then

$$\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x') \tag{3.9}$$

which is called the *reproducing property* of the kernel [Scholkopf and Smola, 2001].

Finally, by combining Eq. 3.8 and 3.9, and applying the Cauchy-Schwartz inequality

$$|\langle x, y \rangle|^2 \leq \langle x, x \rangle \cdot \langle y, y \rangle \,, \tag{3.10}$$

we can prove the desired property

$$\langle k(\cdot, x), f \rangle^2 \leq \langle k(\cdot, x), k(\cdot, x) \rangle \cdot \langle f, f \rangle \qquad \Rightarrow \tag{3.11}$$

$$f(x)^2 \leq k(x, x) \langle f, f \rangle \qquad \Rightarrow \tag{3.12}$$

$$\langle f, f \rangle = 0 \Rightarrow f(x) = 0 \tag{3.13}$$

3.1 The Kernel Trick

We shall now introduce the kernel trick, by considering data in the input space defined by $\boldsymbol{x} \in \mathbb{R}^D$. The motivation behind the kernel trick, is to map data that are non-separable by a linear classifier in the input space, to some high-dimensional space, where data then may be linearly separable. Thus, a separating hyperplane in a high-dimensional space, may correspond to a highly

nonlinear boundary in the input space, which makes linear classification in the new space equivalent to nonlinear classification in the original space.

The Mercer theorem states that any continuous, symmetric, positive semidefinite kernel function, $k(\boldsymbol{x}, \boldsymbol{x}')$, can be expressed as a dot product in a high-dimensional space [Kimeldorf and Wahba, 1970]. Thus, if a kernel satisfies these properties there will exist a respective RKHS, such that

$$\langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}') \rangle = \langle k(\cdot, \boldsymbol{x}), k(\cdot, \boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}'). \tag{3.14}$$

The result implies that without knowing the mapping defined by $\varphi(\cdot)$, the dot product can be computed in the high-dimensional space solely in terms of the respective kernel, which is essential from a computational perspective.

The kernel trick is applicable for any algorithm where data only enters in terms of dot products, as we shall see in Section 3.2. To conclude this section, let us briefly review two commonly used kernels. The linear kernel is given by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}', \tag{3.15}$$

and recovers the dot product in the input space, whereas the commonly used Gaussian kernel is given by

$$k_\gamma(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2), \tag{3.16}$$

where the γ parameter defines the width of the kernel. If the Gaussian is very wide, the kernel approximates the linear kernel, whereas if the Gaussian is very narrow, only nearby points are taken into account, which may lead to overfitting. I refer to [Scholkopf and Smola, 2001] for a discussion on other kernels and their applications.

3.2 Supervised Learning

To see how the kernel trick can be exploited in practice, let us consider the following very general objective[1]

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} \sum_{i=1}^{N} L(y_i, \boldsymbol{w}^T \boldsymbol{x}_i) + \lambda \|\boldsymbol{w}\|_2^2, \tag{3.17}$$

where $\boldsymbol{x} \in \mathbb{R}^D$ and $y \in \{1, -1\}$. The well-known SVM can be obtained by inserting the hinge-loss function $L_{\text{SVM}}(y, t) = \max(0, 1 - yt)$ whereas a logistic-loss function $L_{\text{LR}}(y, t) = \log(1 + \exp(-yt))$ results in a Logistic Regression (LR)

[1] A bias term have been omitted for notational convenience.

model. Moreover, the L_2 regularizer incorporated in the general model shown here, corresponds to a Gaussian prior in a Baysian framework and provides weight shrinkage. A Laplace prior corresponds to $L_1 \sim \|\boldsymbol{w}\|_1 = \sum_i |w_i|$ regularization and will in the Maximum A Posteriori (MAP) estimate provide sparsity which often is an advantage for the interpretability of the solution. Note that marginalization over $\boldsymbol{w}$ will not result in sparsity since the Laplace distribution assigns probability mass outside the mode, hence sparsity comes from the MAP point estimate.

We can apply the kernel trick to Eq. 3.17, by applying the feature space mapping $\varphi : \mathbb{R}^D \mapsto \mathcal{H}$ to the training data, and use the representer theorem to rewrite $\boldsymbol{w} = \sum_{i=1}^N \alpha_i \varphi(\boldsymbol{x}_i)$. By direct substitution this gives

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^N L\left(y_i, \sum_{j=1}^N \alpha_j \varphi(\boldsymbol{x}_j)^T \varphi(\boldsymbol{x}_i)\right) + \lambda \sum_{i,j=1}^N \alpha_i \alpha_j \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j). \tag{3.18}$$

We can now apply the kernel trick, as the training data solely enters as inner products

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^N L\left(y_i, \sum_{j=1}^N \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i)\right) + \lambda \sum_{i,j=1}^N \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{3.19}$$

By introducing the kernel matrix $K_{ij} = k(\boldsymbol{x}_j, \boldsymbol{x}_i)$, the above expression simplifies to

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \sum_{i=1}^N L\left(y_i, \sum_{j=1}^N \boldsymbol{K}_i^T \boldsymbol{\alpha}\right) + \lambda \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}. \tag{3.20}$$

As long as we choose L to be differentiable we can optimize this unconstrained function using gradient descent. Even simple kernel machines scales with $\mathcal{O}(N^3)$, as required by inverting the kernel matrix, and with respect to the SVM, most modern algorithms rely on dual formulations and solve a related Quadratic Programming (QP) with box constraints [Platt, 1999]. Also primal formulations exist, based on differentiable approximations to the hinge loss, *e.g.*, the Huber loss, and empirically the SVM may scale much better than $\mathcal{O}(N^3)$ if only few points define the margin, *i.e.*, few support vectors [Chapelle, 2007]. See also [Muller et al., 2001].

In a Bayesian terminology, the GP is a stochastic process that is defined as a collection of random variables, where any finite subset must have a joint Gaussian distribution. In effect, the GP is placed as a prior over any finite set of functional values $\boldsymbol{f} = [f_1, f_2, ..., f_n]^T$, where $f_i = f(\boldsymbol{x}_i)$, resulting in a finite multivariate Gaussian distribution over the set as $p(\boldsymbol{f}|\boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, where

$\boldsymbol{K}$ is the kernel matrix. The predictive distribution is a immediate advantage of the probabilistic GP framework, since the mean and variance can be used to quantify uncertainty in the predictions made by the model. In [Nielsen et al., 2013] we considered modeling user preferences with respect to the settings of a five-band equalizer for audio processing. We developed a framework based on GP regression and a sequential design method for effectively selecting the next equalizer setting to be evaluated by the subject. For more details see Appendix G.

In general, kernel methods heavily depend on the choice of kernel hyperparameter(s), but most often direct optimization of the these are infeasible due to non-convexities imposed by the kernel. Moreover, a Bayesian treatment such as Automatic Relevance Determination (ARD) will prove computationally heavy even for moderate sized problems when analytic integration over the parameter space is intractable. The default approach is to carry out an exhaustive grid search, at a sufficient resolution, over some predefined range of the parameters, and the best parameters are then found by minimizing the CV error. Obviously, such an approach will only be suitable for small to moderate sized problems, whereas for large problems this strategy will be computationally infeasible. In [Hansen et al., 2011a] that can be found in Appendix B, we propose a novel algorithm for hyperparameter selection where we use a MEB as a measure of the dispersion of cluster means in the RKHS. Our approach is motivated by previous studies on binary classification, demonstrating how the intercluster distance in RKHS and the optimal hyperparameter defining the RKHS correlates [Wu and Wang, 2009; Xiaoshan et al., 2010]. For binary classification, there exist several other computational attractive techniques for finding good parameterizations [Joachims, 2001; Vapnik and Chapelle, 2000; Wahba, 1999], and likewise for multi class problems [Lorena and de Carvalho, 2008; Van Heerden and Barnard, 2010; Varewyck and Martens, 2011]. However, in [Duan et al., 2003] it was shown that all of these approximation schemes were inferior to standard 5-fold CV. Our algorithm in [Hansen et al., 2011a] optimizes the parameterization of the RKHS, by maximizing the MEB that can be regarded as a quality proxy of the RKHS in terms of discriminative properties. A sublinear algorithm for finding the MEB in a finite dimensional input space was introduced by [Clarkson et al., 2010], and in our contribution we generalize their randomized MEB estimation procedure in a RKHS formulation, thereby providing competitive time complexities with respect to existing distance metrics in the RKHS. We demonstrate the developed algorithm by considering image classification on the Amsterdam Library of Object Images (ALOI) [Geusebroek et al., 2005], and the proposed methodology outperforms related distance measures in the RKHS, such as the *median*, *mean*, *maximum* and *minimum*, in that only the MEB approach provides the same parameter estimate as the computationally expensive 5-fold CV procedure.

3.3 The Nyström Approximation

Low-rank matrix decompositions have recently gained popularity in scaling up kernels methods to large amounts of data. The general assumption behind such approximations are low-rankness of data, *e.g.*, the kernel matrix used for encoding the similarity between data samples can often be well-approximated by a few eigenvectors due to a rapidly decaying spectrum [Williams and Seeger, 2000].

The Nyström approximation [Nyström, 1930] is a widely used low-rank approach in the machine learning community, and originated from solving integral equations [Williams and Seeger, 2001]. Let $k(\cdot, \cdot)$ denote the kernel function and let $p(\cdot)$ denote the underlying sample set distribution, then, the Nyström method approximates the following integral equation

$$\int k(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{y}) \phi_i(y) d\boldsymbol{y} = \lambda_i \phi_i(\boldsymbol{x}), \tag{3.21}$$

where $\phi_i(\boldsymbol{x})$ and λ_i corresponds to the i^{th} eigenfunction and eigenvalue of $k(\cdot, \cdot)$. In the vanilla approach the integral is then approximated by sampling n data points that are drawn from the underlying distribution, and using these, we can approximate the expectation with the empirical average

$$\frac{1}{n} \sum_{j=1}^{n} k(\boldsymbol{x}, \boldsymbol{y}_j) \phi_i(\boldsymbol{y}_j) = \lambda_i \phi_i(\boldsymbol{x}). \tag{3.22}$$

This approximation technique requires $\mathcal{O}(nN)$ space and runs in $\mathcal{O}(n^2 N)$, thereby allowing for large-scale kernel machines. The Nyström approximation can be found in various forms, and most techniques differ in the way they sample data. For instance [Gittens and Mahoney, 2013] uses leverage scores to do non-uniform sampling, whereas [Kumar et al., 2009] uses an ensemble approach.

With respect to semi-supervised eigenvectors [Hansen and Mahoney, 2013], we specifically consider how a low-rank decomposition can be exploited to yield very efficient solutions, where the running time of computing a given semi-supervised eigenvector largely depends on a matrix-vector product.

3.4 Kernel PCA

Kernel PCA [Schölkopf et al., 1998] is the natural generalization of linear PCA, a commonly applied unsupervised dimensionality reduction method, that works

by projecting multidimensional data onto the directions of the largest variance, which often facilitates the classification of data [Hotelling, 1933]. This is done by an orthogonal linear transformation that transforms the data to a new coordinate system, such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), whereas the second lies in the direction having the second greatest variance and so forth.

Compared to the traditional formulation, we here adapt an approach similar to [Walder et al., 2010b] and formulate the kernel PCA objective as

$$\max_{f_n \in \mathcal{H}} \quad \sum_{i=1}^{N} \left(f_n(\boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} f_n(\boldsymbol{x}_j) \right)^2 \tag{3.23}$$

$$\text{s.t} \quad \|f_n\|_{\mathcal{H}}^2 = 1 \tag{3.24}$$

$$\sum_{i=1}^{n-1} \langle f_n, f_i \rangle_{\mathcal{H}}^2 = 0, \tag{3.25}$$

where $\mathcal{H}$ is the RKHS defined by the kernel, and f_n the n^{th} solution. We can then apply the representer theorem $f^\star(\cdot) = \sum_{i=1}^{N} \alpha_i^\star k(\boldsymbol{x}_i, \cdot)$ and form the derivative with respect to $\boldsymbol{\alpha}$ of the Lagrangian, leading to the following generalized eigenvalue problem

$$\boldsymbol{K}\boldsymbol{\alpha} = \lambda(\boldsymbol{K}^T\boldsymbol{K} - \boldsymbol{K}^T\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{\alpha}, \tag{3.26}$$

where $\boldsymbol{E}_N$ is a matrix of size N with entries $\frac{1}{N}$. This solution is equivalent up to a renormalization of the original problem [Schölkopf et al., 1998] in its centered form, and the orthogonality constraint in the above formulation is implicitly satisfied in the solution given by the generalized eigenvectors.

In [Hansen et al., 2013] we generalize the work of [Walder et al., 2010b], who developed an optimization ansatz for a single semi-supervised kernel PCA component. Our technique in [Hansen et al., 2013] elaborates on the previous work in that we provide a methodology to construct multiple semi-supervised kernel PCA components, that all are biased by labeled data. Our formulation gives rise to the following objective, that can be regarded as a constrained eigenvalue

problem

$$\max_{f_n \in \mathcal{H}} \quad \sum_{i=1}^{N} \left(f_n(\boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} f_n(\boldsymbol{x}_j) \right)^2 \tag{3.27}$$

$$\text{s.t} \quad \|f_n\|_{\mathcal{H}}^2 = 1 \tag{3.28}$$

$$\sum_{i=1}^{n-1} \langle f_n, f_i \rangle_{\mathcal{H}}^2 = 0 \tag{3.29}$$

$$\sum_{i \in \mathcal{L}} (f_n(\boldsymbol{x}_i) - y_i)^2 \leq \omega, \tag{3.30}$$

in that it extends the usual kernel PCA objective with a least squares loss term. Here $\mathcal{L}$ is the set of labeled training data and ω determines the allowed derivation from the true labels. The general aim of this methodology is to exploit the class labels of data to bias the manifold representation in such a way that the obtained kernel PCA basis is "true" to the labels. Constrained eigenvalue problems, are in general difficult to solve [Gander et al., 1989b], but as we show in [Hansen et al., 2013] it turns that by computing the components sequentially, conditioned on being perpendicular to the previous components, the loss term may be saturated by using a binary search that exploits a monotonic relationship between the actual loss and the governing Lagrange multiplier. In particular, the semi-supervised kernel PCA components were in [Hansen et al., 2013] used to facilitate image denoising through pre-image estimation.

3.5 Denoising through Pre-image Estimation

In denoising applications kernel PCA provides the basis for dimensionality reduction, prior to the so-called pre-image problem where denoised feature space points, as represented by a low-rank structure, are mapped back to input space. Inherently, the main obstacle is that the problem is ill-posed due to the non-bijective feature space mapping. The pre-image problem of reconstructing kernel PCA projections in input space have been faced in a variety of ways, most of which are limited to a specific choice of kernel embedding, see for instance [Bakir et al., 2004; Dambreville et al., 2006; Kwok and Tsang, 2003; Mika et al., 1999].

By the representer theorem, the projection of a feature space mapped test point onto the n^{th} principal component is

$$\beta_n(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_{ni} k_c(\boldsymbol{x}, \boldsymbol{x}_i) \tag{3.31}$$

where k_c is the centered kernel, and for convenience we will in the following just write β_n as $\boldsymbol{x}$ will be implicit from the context. The projection of $\varphi(\boldsymbol{x})$ onto the subspace spanned by the leading q components can be written as

$$P_q\varphi(\boldsymbol{x}) = \sum_{n=1}^{q} \beta_n \sum_{i=1}^{N} \alpha_{ni}\varphi_c(\boldsymbol{x}_i) + \bar{\varphi} \tag{3.32}$$

where $\bar{\varphi} = \frac{1}{N}\sum_{i=1}^{N}\varphi(\boldsymbol{x}_i)$ is the mean of the φ-mapped data points and $\varphi_c(\boldsymbol{x}_i) = \varphi(\boldsymbol{x}_i) - \bar{\varphi}$ is the centered feature space mapping of $\boldsymbol{x}$ [Schölkopf et al., 1998].

The ill-posed pre-image problem can be relaxed to that of finding an approximate pre-image, *i.e.*, a point in input space, say $\boldsymbol{z}$, that maps into a point in feature space "as close as possible" to $P_q\varphi(\boldsymbol{x})$. Hence, we can think of the problem as a search where we seek to minimize the distance in the RKHS between $\varphi(\boldsymbol{z})$ and $P_q\varphi(\boldsymbol{x})$ with respect to $\boldsymbol{z}$ [Mika et al., 1999]. Using the kernel trick for the squared error, we get

$$\rho = ||\varphi(\boldsymbol{z}) - P_q\varphi(\boldsymbol{x})||^2 = k(\boldsymbol{z},\boldsymbol{z}) - 2\sum_{n=1}^{N}\xi_n k(\boldsymbol{z},\boldsymbol{x}_n) + \Omega \tag{3.33}$$

where all the $\boldsymbol{z}$-independent terms are collected in Ω, and $\xi_n = \tilde{\xi}_n + \frac{1}{N}(1 - \sum_{j=1}^{N}\tilde{\xi}_j)$, with $\tilde{\xi}_n = \sum_{i=1}^{q}\beta_i\alpha_{in}$. In extrema, the derivative with respect to $\boldsymbol{z}$ is zero, giving rise to the following fixed-point iteration for the Gaussian kernel [Mika et al., 1999]

$$\boldsymbol{z}_{t+1} = \frac{\sum_{n=1}^{N}\xi n \exp(-\gamma||\boldsymbol{z}_t - \boldsymbol{x}_n||^2)\boldsymbol{x}_n}{\sum_{n=1}^{N}\xi n \exp(-\gamma||\boldsymbol{z}_t - \boldsymbol{x}_n||^2)} = \frac{[\boldsymbol{\xi}\circ\boldsymbol{k}_{\boldsymbol{z}_t}]^T\boldsymbol{X}}{\boldsymbol{\xi}^T\boldsymbol{k}_{\boldsymbol{z}_t}} \tag{3.34}$$

The cost in Equation (3.33) may be highly multi modal, leading to a non-convex optimization problem, so the fixed-point iteration scheme can suffer from convergence to local minima.

As discussed in Section 3.4, the main motivation behind [Hansen et al., 2013], for introducing semi-supervised kernel PCA, was to facilitate denoising when side-information was available. Besides introducing a generalization of semi-supervised kernel PCA, [Hansen et al., 2013] also introduces a method for exploiting intrinsic structures in data, to improve the pre-images. Specifically, we device a fixed-point iteration for the kernel proposed by [Sindhwani et al., 2005], that is based on the idea of warping the RKHS to account for the manifold structure imposed by both labeled and unlabeled data points. The kernel exploits the smoothness of data as modeled by the combinatorial graph Laplacian $\boldsymbol{L}_G$, and is given by

$$\tilde{k}(\boldsymbol{x},\boldsymbol{y}) = k(\boldsymbol{x},\boldsymbol{y}) - \boldsymbol{k}_x^T(\boldsymbol{I} + \boldsymbol{L}_G\boldsymbol{K})^{-1}\boldsymbol{L}_G\boldsymbol{k}_y, \tag{3.35}$$

where $\boldsymbol{k_y} = [k(\boldsymbol{y}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{y}, \boldsymbol{x}_N)]^T$. Details regarding the properties of graph Laplacians and related methods will be discussed in Chapter 4. Regarding the fixed-point iteration for this kernel, we choose for simplicity as well as scalability not to include the pre-image in $\boldsymbol{K}$, thereby avoiding the inversion of $(\boldsymbol{I} + \boldsymbol{L}_G\boldsymbol{K})^{-1}$ at every iteration. Note that the effect of this relaxation is only minor if the manifold is well defined by the training samples.

Letting $\boldsymbol{M} \stackrel{\text{def}}{=} (\boldsymbol{I} + \boldsymbol{L}_G\boldsymbol{K})^{-1}\boldsymbol{L}_G$, the kernel simplifies to $\tilde{k}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{k}_x^\top \boldsymbol{M}\boldsymbol{k}_y$, and by inserting this new kernel into the cost function in Eq. 3.33, we obtain the following fixed-point iteration

$$\begin{aligned}\boldsymbol{z}_{t+1} = &\frac{\left[(\boldsymbol{M} \circ (\boldsymbol{k}_{\boldsymbol{z}_t}\boldsymbol{k}_{\boldsymbol{z}_t}^T - \boldsymbol{k}_{\boldsymbol{z}_t}(\boldsymbol{K}\boldsymbol{\xi})^T - (\boldsymbol{K}\boldsymbol{\xi})\boldsymbol{k}_{\boldsymbol{z}_t}^T))\mathbf{1}\right]^T \boldsymbol{X}}{(\boldsymbol{k}_{\boldsymbol{z}_t}^T\boldsymbol{M} + \boldsymbol{\xi}^T - 2\boldsymbol{\xi}^T\boldsymbol{K}\boldsymbol{M})\boldsymbol{k}_{\boldsymbol{z}_t}} \\ &+ \frac{[\boldsymbol{M} \circ \boldsymbol{\xi} \circ \boldsymbol{k}_{\boldsymbol{z}_t}]^T \boldsymbol{X}}{(\boldsymbol{k}_{\boldsymbol{z}_t}^T\boldsymbol{M} + \boldsymbol{\xi}^T - 2\boldsymbol{\xi}^T\boldsymbol{K}\boldsymbol{M})\boldsymbol{k}_{\boldsymbol{z}_t}}\end{aligned} \tag{3.36}$$

The introduced methodologies was in [Hansen et al., 2013] evaluated on both synthetic and image data from the ALOI database [Geusebroek et al., 2005]. For the images we constructed a denoising problem by randomly adding two images from the database, where one image had half the intensity of the other, and the goal was to reconstruct the dominant image. Our method based on semi-supervised kernel PCA and the graph regularized kernel [Sindhwani et al., 2005], produced significantly better reconstructions, and proved to be less sensitive to the choice of kernel parameter than the usual approach based on classical kernel PCA and the fixed-point iteration by [Mika et al., 1999].

Chapter 4

Spectral Methods

This chapter introduces spectral clustering as a general motivation for the field of spectral methods, and we then discuss key results with respect to the two major contributions of this thesis [Hansen and Mahoney, 2012, 2013] that introduces the concept of semi-supervised eigenvectors of a graph Laplacian.

Spectral techniques have in recent years become popular in machine learning tasks such as image segmentation [Malik, 2000], clustering [Fowlkes et al., 2004], semi-supervised learning [Zhu, 2005], and most recently for providing closed-form solutions to various existing methods such as Latent Dirichlet Allocation (LDA) and Gaussian Mixture Models (GMM) [Anandkumar et al., 2012].

In the context of spectral graph theory the main structure is the *graph Laplacian* which, at first glance, may seem mysterious due to its remarkable clustering abilities. Spectral clustering serves as a good introduction to the general field of spectral methods, so before we dive into related details let us first define the general notation.

Given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and some notation of similarity $w_{ij} \geq 0$ between all pairs of data points, the intuitive goal of clustering is to divide the points into groups such that similar points are grouped together. A common approach is to represent such similarities in the form of similarity graphs; *i.e.*, let $G = (V, E, w)$ be a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges, in which edge $\{i, j\}$ has weight w_{ij}. In the following, $\boldsymbol{A}_G \in \mathbb{R}^{V \times V}$ will denote the adjacency matrix of G, while $\boldsymbol{D}_G \in \mathbb{R}^{V \times V}$ will denote the diagonal degree matrix of G, *i.e.*, $\boldsymbol{D}_G(i, i) = d_i = \sum_{j=1}^{n} w_{ij}$ is the weighted degree of vertex i. Moreover, for a set of vertices $S \subseteq V$ in a graph, the *volume of* S is $\mathrm{vol}(S) \stackrel{\text{def}}{=} \sum_{i \in S} d_i$.

The clustering problem may now be reformulated in terms of the similarity graph, *i.e.*, a clustering is a partitioning of the graph such that edges between different groups have low weights, whereas points within a group have high weights. Obviously the way we measure similarity is crucial for the representation of latent clusterings, so the similarity graph must be a good model for representing local heterogeneities. Three popular and common approaches for constructing a similarity graph are briefly discussed below:

- *The ϵ-neighborhood graph:* Connects all points whose pairwise distances are smaller than ϵ. These sorts of graphs are usually considered as unweighted graphs as the distances between all connected points are roughly of equal scale.

- *The k-nearest neighbor graph:* Connects vertex v_i with v_j if v_j is among the k-nearest neighbors of v_i. Such a definition leads to a directed graph that is not symmetric. The graph can be made undirected in various ways; one approach is to ignore the direction of the edges by connecting v_i with v_j with an undirected edge if v_i is among the k-nearest neighbors of v_j, or if v_j is among the k-nearest neighbors of v_i. All vertices in such a graph will therefore have at least k neighbors and at most $n - 1$, ensuring

that the resulting graph is connected. A second strategy is to connect vertices v_i and v_j if both v_i is among the k-nearest neighbors of v_j, and v_j is among the k-nearest neighbors of v_i. Compared to the first strategy, usually referred to as a *k-nearest neighbor graph*, the resulting graph of the latter strategy is denoted a *mutual k-nearest neighbor graph* and is not necessarily connected.

- *The fully connected graph:* Connects all points with a non-negative similarity w_{ij}. A popular similarity measure is the Gaussian kernel $k_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$ as introduced in Section 3.1, where γ is a hyperparameter used for adjusting the width of the neighborhood.

With these graph definitions in mind we now consider the main tools of spectral clustering, namely *Laplacian matrices*. As the spectral properties of such matrices are the key ingredient in spectral clustering let us first consider some of the properties of what is known as the *combinatorial graph Laplacian*. Note that, when we refer to "the leading eigenvector" of Laplacian matrices we implicitly mean the eigenvector with the smallest eigenvalue, and moreover, do we assume eigenvalues to be ordered increasingly.

The combinatorial graph Laplacian is defined as

$$\boldsymbol{L}_G \overset{\text{def}}{=} \boldsymbol{D}_G - \boldsymbol{A}_G, \tag{4.1}$$

and poses several interesting properties. For example, for every vector $\boldsymbol{x} \in \mathbb{R}^n$ we have

$$\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2. \tag{4.2}$$

To see why this is true we expand the expression and exploit the definition $d_i = \sum_{j=1}^{n} w_{ij}$

$$\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{A}_G \boldsymbol{x} = \sum_{i=1}^{n} d_i x_i^2 - \sum_{i,j=1}^{n} x_i x_j w_{ij} \tag{4.3}$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} d_i x_i^2 + \sum_{j=1}^{n} d_j x_j^2 - 2 \sum_{i,j=1}^{n} x_i x_j w_{ij} \right) \tag{4.4}$$

$$= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2. \tag{4.5}$$

From the definition of $\boldsymbol{L}_G$ and the above relationship it is evident that $\boldsymbol{L}_G$ is positive semidefinite (PSD), and that the smallest eigenvalue of $\boldsymbol{L}_G$, with

respect to the generalized eigenvalue problem $\boldsymbol{L}_G\boldsymbol{v} = \lambda\boldsymbol{D}_G\boldsymbol{v}$, is $\lambda_1 = 0$, corresponding to the constant vector $\boldsymbol{v} = \mathbf{1}$. Yet an interesting property is that the number of connected components in G is directly related to the spectrum of $\boldsymbol{L}_G$, as the multiplicity of the eigenvalue 0 of $\boldsymbol{L}_G$ equals the number of connected components, see [Von Luxburg, 2007].

4.1 Spectral Clustering

To see how the graph Laplacian is related to clustering, let us start by defining the value of a "cut" in the graph as

$$\text{cut}(A, B) \stackrel{\text{def}}{=} \sum_{i\in A, j\in B} w_{ij}. \tag{4.6}$$

Here A and B defines a potential partitioning of the vertices in the graph and the cost of that partitioning is measured by the sum of edge weights connecting vertices in A to vertices in B. The "normalized cut", introduced in [Malik, 2000], incorporates an extra term to favor balanced partitions

$$\text{Ncut}(A, B) \stackrel{\text{def}}{=} \sum_{i\in A, j\in B} w_{ij} \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right). \tag{4.7}$$

Such combinatorial clustering problems are NP-hard and spectral clustering refers to the continuous relaxation of such problems. To derive such a continuous relaxation for Eq. 4.7 we define a cluster indicator vector as follows

$$x_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ \frac{-1}{\text{vol}(B)} & \text{if } i \in B. \end{cases} \tag{4.8}$$

By inserting this definition into Eq. 4.5 we get

$$\boldsymbol{x}^T\boldsymbol{L}_G\boldsymbol{x} \propto \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2 = \sum_{i\in A, j\in B} w_{ij} \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)} \right)^2. \tag{4.9}$$

Similarly we can apply $\boldsymbol{x}$ on both sides of $\boldsymbol{D}_G$

$$\boldsymbol{x}^T\boldsymbol{D}_G\boldsymbol{x} = \sum_{i}^{n} d_i x_i^2 = \sum_{i\in A} \frac{d_i}{\text{vol}(A)^2} + \sum_{i\in B} \frac{d_i}{\text{vol}(B)^2} = \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}. \tag{4.10}$$

We now see that by combining Eq. 4.9 and 4.10 we arrive at the normalized cut defined in Eq. 4.7

$$\text{Ncut}(A, B) \propto \frac{\boldsymbol{x}^T\boldsymbol{L}_G\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{D}_G\boldsymbol{x}}, \tag{4.11}$$

that we now recognize as the generalized Rayleigh quotient [Golub and Van Loan, 2012]. Hence, we can minimize the righthand side of the above expression, by relaxing $\boldsymbol{x}$ to take real values, and then choose $\boldsymbol{x}$ to be the eigenvector having the second smallest eigenvalue of the generalized eigenvalue problem

$$\boldsymbol{L}_G\boldsymbol{v} = \lambda \boldsymbol{D}_G\boldsymbol{v}. \tag{4.12}$$

The reason for choosing the second smallest is that the smallest eigenvalue, $\lambda_1 = 0$ as deduced from Eq. 4.5, gives rise to the constant eigenvector $\boldsymbol{v}_1 = \mathbf{1}$ that is infeasible with respect to the normalized cut [Malik, 2000]. With respect to $\boldsymbol{v}_2$, thresholding is usually applied as a post-processing step to convert back to discrete assignments used for partitioning the graph. Also, in practical applications, where we are interested in multiple clusterings, a usual approach is to compute a series of the smallest eigenvectors, as these captures the slowest modes of variation in increasing order, and then perform clustering in that embedding, using *e.g.*, k-means [Von Luxburg, 2007].

Finally, by substituting with $\boldsymbol{v} = \boldsymbol{D}_G^{-1/2}\boldsymbol{z}$ in the generalized eigenvalue problem in Eq. 4.12, we obtain the standard eigenvalue problem

$$\mathcal{L}_G\boldsymbol{z} = \lambda \boldsymbol{z}, \tag{4.13}$$

where $\mathcal{L}_G$ is the *normalized graph Laplacian* given by

$$\mathcal{L}_G \stackrel{\text{def}}{=} \boldsymbol{D}_G^{-1/2}\boldsymbol{L}_G\boldsymbol{D}_G^{-1/2} = \boldsymbol{I} - \boldsymbol{D}_G^{-1/2}\boldsymbol{A}_G\boldsymbol{D}_G^{-1/2}. \tag{4.14}$$

4.2 Semi-supervised Eigenvectors

In many applications, one has information about a specific target region of a large graph, *e.g.*, that is provided in a semi-supervised manner, and one wants to perform common machine learning and data analysis tasks "near" the pre-specified target region. For example, in social and information network analysis, one might have a few "ground truth" nodes that belong to a cluster or community of interest, and one might want to perform link prediction or to refine the cluster in order to find other nearby members. In computer vision, one might have a large corpus of images along with a set of pixels provided by a face detection algorithm, and one might want to segment entire heads from the background for all the images in the corpus in an automated manner. In fMRI applications, one might have sets of neurons that "fire" in response to some external experimental stimulus, and one might want to analyze the subsequent temporal dynamics of neuron stimulation.

Hence, in terms of such "local" side-information we are interested in doing *locally-biased machine learning*, meaning that we have a dataset, *e.g.*, represented as a graph, and that we have information, *e.g.*, given in a semi-supervised manner, that certain "regions" of of the data graph are of particular interest. In this case, we may want to focus predominantly on those regions and perform data analysis and machine learning, *e.g.*, classification, clustering, ranking, *etc.*, that is "biased towards" those pre-specified regions. Examples of this include the following.

- *Locally-biased community identification.* In social and information network analysis, one might have a small "seed set" of nodes that belong to a cluster or community of interest [Andersen and Lang, 2006; Leskovec et al., 2008]; in this case, one might want to perform link or edge prediction, or one might want to "refine" the seed set in order to find other nearby members.
- *Locally-biased image segmentation.* In computer vision, one might have a large corpus of images along with a "ground truth" set of pixels as provided by a face detection algorithm [Eriksson et al., 2007; Mahoney et al., 2012; Maji et al., 2011]; in this case, one might want to segment entire heads from the background for all the images in the corpus in an automated manner.
- *Locally-biased neural connectivity analysis.* In fMRI applications, one might have small sets of neurons that "fire" in response to some external experimental stimulus [Norman et al., 2006a]; in this case, one might want to analyze the subsequent temporal dynamics of stimulation of neurons that are "nearby," either in terms of connectivity topology or functional response, members of the original set.

In each of these examples, the data is modeled by a graph, which is either given from the application domain, or constructed from feature vectors obtained from the application domain. Hence, one has information that can be viewed as semi-supervised in the sense that it consists of exogeneously-specified "labels" for the nodes of the graph. In addition, there are typically a relatively-small number of labels and one is interested in obtaining insight about the data graph nearby those labels.

Although eigenvector-based methods have received attention in a wide range of machine learning and data analysis applications in recent years, for example; in nonlinear dimensionality reduction [Belkin and Niyogi, 2003; Coifman et al., 2005]; in kernel-based machine learning [Schölkopf and Smola, 2001]; in Nyström-based learning methods [Talwalkar and Rostamizadeh, 2010; Williams and Seeger, 2001]; spectral partitioning [Ng et al., 2001; Pothen et al., 1990;

Shi and Malik, 2000], *etc.*, the above examples present considerable challenges for standard global spectral techniques and other traditional eigenvector-based methods. At root, the reason is that eigenvectors are inherently global quantities, limiting their applicability in situations where one is interested in very local properties of the data. That is, very local information can be "washed out" and essentially become invisible to these globally-optimal vectors. For example, a sparse cut in a graph may be poorly correlated with the second eigenvector and thus invisible to a method based only on eigenvector analysis. Hence, if one has semi-supervised information about a specific target region in the graph, as in the above examples, one might be interested in finding clusters near this pre-specified local region in a semi-supervised manner.

In the remainder of this section, we present our contributions [Hansen and Mahoney, 2012] and [Hansen and Mahoney, 2013] that can be found in Appendix G and F, respectively. The original algorithm of [Mahoney et al., 2012] introduced a methodology to construct a locally-biased version of the leading nontrivial eigenvector of a graph Laplacian and showed (theoretically and empirically in a social network analysis application) that the resulting vector could be used to partition a graph in a locally-biased manner. From this perspective, our extension incorporates a natural orthogonality constraint that successive vectors need to be orthogonal to previous vectors. Subsequent to the work of [Mahoney et al., 2012], [Maji et al., 2011] applied the algorithm of [Mahoney et al., 2012] to the problem of finding locally-biased cuts in a computer vision application. Similar ideas have also been applied somewhat differently. For example, [Andersen and Lang, 2006] use locally-biased random walks, *e.g.*, short random walks starting from a small seed set of nodes, to find clusters and communities in graphs arising in Internet advertising applications; [Leskovec et al., 2008] used locally-biased random walks to characterize the local and global clustering structure of a wide range of social and information networks; [Joachims, 2003] developed the Spectral Graph Transducer (SGT), that performs transductive learning via spectral graph partitioning. The objectives in both [Joachims, 2003] and [Mahoney et al., 2012] are considered constrained eigenvalue problems, that can be solved by finding the smallest eigenvalue of an asymmetric generalized eigenvalue problem, but in practice this procedure can be highly unstable [Gander et al., 1989a]. The SGT reduces the instabilities by performing all calculations in a subspace spanned by the d smallest eigenvectors of the graph Laplacian, whereas [Mahoney et al., 2012] performs a binary search, exploiting the monotonic relationship between a control parameter and the corresponding Lagrange multiplier.

Although the GlobalSpectral objective, presented on the left of Figure 4.1, is a non-convex optimization problem, strong duality holds for it and the solution may be computed as $\boldsymbol{v}_2$, *i.e.*, the leading nontrivial generalized eigenvector of $\boldsymbol{L}_G$. The next eigenvector $\boldsymbol{v}_3$ is the solution to GlobalSpectral, augmented

GlobalSpectral

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} \\ \text{s.t} \quad & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{x} = 1 \\ & \boldsymbol{x}^T \boldsymbol{D}_G \mathbf{1} = 0 \end{aligned}$$

LocalSpectral

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} \\ \text{s.t} \quad & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{x} = 1 \\ & \boldsymbol{x}^T \boldsymbol{D}_G \mathbf{1} = 0 \\ & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{s} \geq \sqrt{\kappa} \end{aligned}$$

Generalized LocalSpectral

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} \\ \text{s.t} \quad & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{x} = 1 \\ & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{X} = 0 \\ & \boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{s} \geq \sqrt{\kappa} \end{aligned}$$

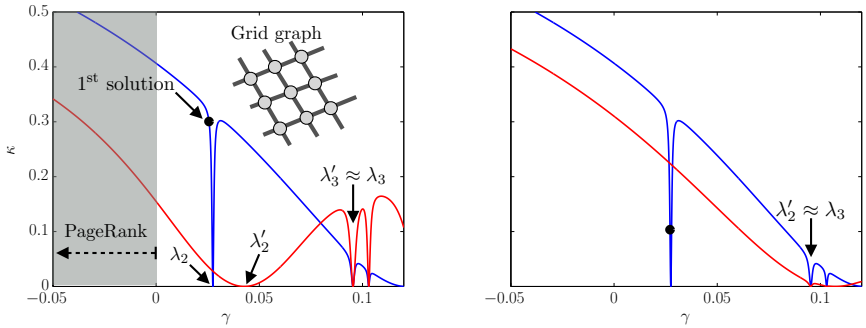
Figure 4.1: Left: The usual GlobalSpectral partitioning optimization problem; the vector achieving the optimal solution is $\boldsymbol{v}_2$, the leading nontrivial generalized eigenvector of $\boldsymbol{L}_G$ with respect to $\boldsymbol{D}_G$. Middle: The LocalSpectral optimization problem, which was originally introduced in [Mahoney et al., 2012]; for $\kappa = 0$, this coincides with the usual global spectral objective, while for $\kappa > 0$, this produces solutions that are biased toward the seed vector $\boldsymbol{s}$. Right: The Generalized LocalSpectral optimization problem we introduce that includes both the locality constraint and a more general orthogonality constraint. Our main algorithm for computing semi-supervised eigenvectors will iteratively compute the solution to Generalized LocalSpectral for a sequence of $\boldsymbol{X}$ matrices. In all three cases, the optimization variable is $\boldsymbol{x} \in \mathbb{R}^n$.

with the constraint that $\boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{v}_2 = 0$; and in general the t^{th} generalized eigenvector of $\boldsymbol{L}_G$ is the solution to GlobalSpectral, augmented with the constraints that $\boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{v}_i = 0$, for $i \in \{2, \ldots, t-1\}$. Clearly, this set of constraints and the constraint $\boldsymbol{x}^T \boldsymbol{D}_G \mathbf{1} = 0$ can be written as $\boldsymbol{x}^T \boldsymbol{D}_G \boldsymbol{X} = \mathbf{0}$, where $\mathbf{0}$ is a $(t-1)$-dimensional all-zeros vector, and where $\boldsymbol{X}$ is an $n \times (t-1)$ orthogonal matrix whose i^{th} column equals $\boldsymbol{v}_i$, where $\boldsymbol{v}_1 = \mathbf{1}$, the all-ones vector, is the first column of $\boldsymbol{X}$.

Also presented in Figure 4.1 is LocalSpectral, which includes a constraint requiring the solution to be well-correlated with an input seed set. This LocalSpectral optimization problem was introduced in [Mahoney et al., 2012], where it was shown that the solution to LocalSpectral may be interpreted as a locally-biased version of the second eigenvector of the Laplacian. Although LocalSpectral is non-convex, the solution can be computed efficiently as the solution to a set of linear equations that generalize the popular Personalized PageRank procedure.

Our generalization [Hansen and Mahoney, 2012], Generalized LocalSpectral in Figure 4.1, asks us to find a vector $\boldsymbol{x} \in \mathbb{R}^n$ that minimizes the variance $\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x}$ subject to several constraints; that $\boldsymbol{x}$ is unit length; that $\boldsymbol{x}$ is orthogonal to the span of $\boldsymbol{X}$; and that $\boldsymbol{x}$ is $\sqrt{\kappa}$-correlated with the input seed set vector $\boldsymbol{s}$.

That is, to compute the first semi-supervised eigenvector, we let $\boldsymbol{X} = \mathbf{1}$, *i.e.*, the n-dimensional all-ones vector, which is the trivial eigenvector of $\boldsymbol{L}_G$, and to compute each subsequent semi-supervised eigenvector, we let the columns of $\boldsymbol{X}$ consist of $\mathbf{1}$ and the other semi-supervised eigenvectors found in each of the previous iterations.



(a) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.3$ (b) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.1$

Figure 4.2: For a 2-dimensional grid graph the figures visualize the interplay between the γ parameter and the resulting correlation κ that a semi-supervised eigenvector will end up having with the seed $\boldsymbol{s}$. The general problem is non-convex, but in the range $\gamma \in (-\mathrm{vol}(G), \lambda_2(G))$, there happen to be a monotonic relationship, that we exploit to find the correct setting of γ very efficiently through a binary search. In both figures the blue curve shows the correlation decay as a function of γ for the leading semi-supervised eigenvector, whereas the red shows the decay for the second, conditioned on being perpendicular on the first solution as marked by the black dot, *i.e.*, the second solution will be in the interval $\gamma \in (-\mathrm{vol}(G), \lambda_2'(G))$. In 4.2(a) the leading solution is not too close to λ_2, corresponding to $\boldsymbol{v}_2$, so λ_2' ends up in-between λ_2 and λ_3, whereas in 4.2(b) the leading solution almost coincides with $\boldsymbol{v}_2$, and so $\lambda_2' \approx \lambda_3$, as required if the task is to reproduce the entire sequence of global eigenvectors. Finally, 4.2(a) also highlights the range in which Personalized PageRank can be used for computing solutions to semi-supervised eigenvectors.

As shown in [Hansen and Mahoney, 2012] we can compute the leading semi-supervised eigenvector as the solution to

$$\boldsymbol{x}_1^* = c\,(\boldsymbol{L}_G - \gamma_1 \boldsymbol{D}_G)^+ \boldsymbol{D}_G s, \tag{4.15}$$

where c is chosen to fulfill the norm constraint, and where γ_1 is a Lagrange

multiplier which we can choose to saturate the correlation constraint. In particular, the KKT conditions state that to find the correct setting of γ_1, it suffices to perform a binary search over the interval $\gamma_1 \in (-\text{vol}(G), \lambda_2(G))$ until the correlation constraint is satisfied, *i.e.*, until $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2$ is sufficiently close to κ_1.

To compute subsequent semi-supervised eigenvectors, *i.e.*, at steps $t = 2, \ldots, k$ if one ultimately wants a total of k semi-supervised eigenvectors, then one lets $\boldsymbol{X}$ be the $n \times t$ matrix of the form

$$\boldsymbol{X} = [\boldsymbol{1}, \boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{t-1}^*], \tag{4.16}$$

where $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{t-1}^*$ are successive semi-supervised eigenvectors. The t^{th} semi-supervised eigenvector then takes the form

$$\boldsymbol{x}_t^* = c\left(\boldsymbol{F}\boldsymbol{F}^T(\boldsymbol{L}_G - \gamma_t \boldsymbol{D}_G)\boldsymbol{F}\boldsymbol{F}^T\right)^+ \boldsymbol{D}_G \boldsymbol{s}, \tag{4.17}$$

where the projection matrix $\boldsymbol{F}\boldsymbol{F}^T$ is given by

$$\boldsymbol{F}\boldsymbol{F}^T = \boldsymbol{I} - \boldsymbol{D}_G \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{D}_G \boldsymbol{D}_G \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{D}_G, \tag{4.18}$$

due to the the degree-weighted inner norm. As for the leading semi-supervised eigenvector, the value of γ_t is found through a binary search, but due to the orthogonality constraint, the upper bound for the search is increasing and the value is given by the second smallest eigenvalue of the generalized eigenvalue problem

$$\boldsymbol{F}\boldsymbol{F}^T \boldsymbol{L}_G \boldsymbol{F}\boldsymbol{F}^T \boldsymbol{v}' = \lambda' \boldsymbol{F}\boldsymbol{F}^T \boldsymbol{D}_G \boldsymbol{F}\boldsymbol{F}^T \boldsymbol{v}'. \tag{4.19}$$

The semi-supervised eigenvectors can be thought of as a generalization of the global eigenvectors, *i.e.*, the first solution coincides with the global eigenvector $\boldsymbol{v}_2$ for $\gamma \to \lambda_2$. This is explained by the fact that $(\boldsymbol{L}_G - \gamma \boldsymbol{D}_G)^+ \boldsymbol{D}_G \boldsymbol{s}$ can be interpreted as the first step of the generalized Rayleigh quotient iteration, where γ is the estimate of the eigenvalue, and $\boldsymbol{D}_G \boldsymbol{s}$ is the estimate of the eigenvector. Given that the estimate of the eigenvalue is right, this algorithm will in the initial step compute the corresponding eigenvector. The relationship is visualized in Figure 4.2 that considers the interplay between the γ parameter and the resulting correlation with the seed $\boldsymbol{s}$. For more details see [Hansen and Mahoney, 2013] in Appendix F.

4.2.1 Relationship with Personalized PageRank

The usual interpretation of PageRank involves "random walkers" who uniformly (or non-uniformly, in the case of Personalized PageRank) "teleport" with a probability $\alpha \in (0, 1)$. As visualized in Figure 4.2, a Personalized PageRank procedure can be used to compute solutions to semi-supervised eigenvectors for

$\gamma \in (-\infty, 0)$, corresponding to choosing $\alpha \in (0, 1)$. To make this relationship explicit we rearrange Eq. 4.15 as follows

$$\begin{aligned} \boldsymbol{x}^* &= (\boldsymbol{L}_G - \gamma \boldsymbol{D}_G)^+ \boldsymbol{D}_G \boldsymbol{s} && (4.20)\\ &= c\left((\boldsymbol{D}_G - \boldsymbol{A}_G) - \gamma \boldsymbol{D}_G\right)^+ \boldsymbol{D}_G \boldsymbol{s} && (4.21)\\ &= \frac{c}{1-\gamma}\left(\boldsymbol{D}_G - \frac{1}{1-\gamma}\boldsymbol{A}_G\right)^+ \boldsymbol{D}_G \boldsymbol{s} && (4.22)\\ &= \frac{c}{1-\gamma}\boldsymbol{D}_G^{-1}\left(I - \frac{1}{1-\gamma}\boldsymbol{A}_G\boldsymbol{D}_G^{-1}\right)^+ \boldsymbol{D}_G \boldsymbol{s}, && (4.23) \end{aligned}$$

and we recognize $\boldsymbol{A}_G\boldsymbol{D}_G^{-1}$ as the standard random walk matrix. Hence, it becomes immediate that the solution based on random walkers,

$$\boldsymbol{x}^* = \frac{c}{1-\gamma}\boldsymbol{D}_G^{-1}\left(\boldsymbol{I} + \sum_{i=1}^{\infty}\left(\frac{1}{1-\gamma}\boldsymbol{D}_G^{-1}\boldsymbol{A}_G\right)^i\right)\boldsymbol{D}_G\boldsymbol{s}, \tag{4.24}$$

is divergent for $\gamma > 0$. Since $\gamma = \lambda_2(G)$ corresponds to no regularization and $\gamma \to -\infty$ corresponds to heavy regularization, viewing this problem in terms of solving a linear equation is formally more powerful than viewing it in terms of random walkers. That is, while all possible values of the regularization parameter—and in particular the (positive) value $\lambda_2(\cdot)$—are achievable algorithmically by solving a linear equation, only values in $(-\infty, 0)$ are achievable by running a PageRank diffusion. However in terms of scalability, the PageRank approach is very efficient, and so in [Hansen and Mahoney, 2013] we generalize consecutive semi-supervised eigenvectors in a way that is compatible with the PageRank formulation, by showing that the t^{th} solution can be approximated as

$$\boldsymbol{x}_t^* \approx c\left(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{D}_G\right)(\boldsymbol{L}_G - \gamma_t\boldsymbol{D}_G)^+\boldsymbol{D}_G\boldsymbol{s}, \tag{4.25}$$

under the assumption that all γ_k for $1 < k \leq t$ are sufficiently apart. Hence, the trick is to decouple the projection operator $\boldsymbol{F}\boldsymbol{F}^T$ from the inverse; then use a PageRank procedure to solve Eq. 4.15; and finally apply the projection operator $\left(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{D}_G\right)$ to that solution.

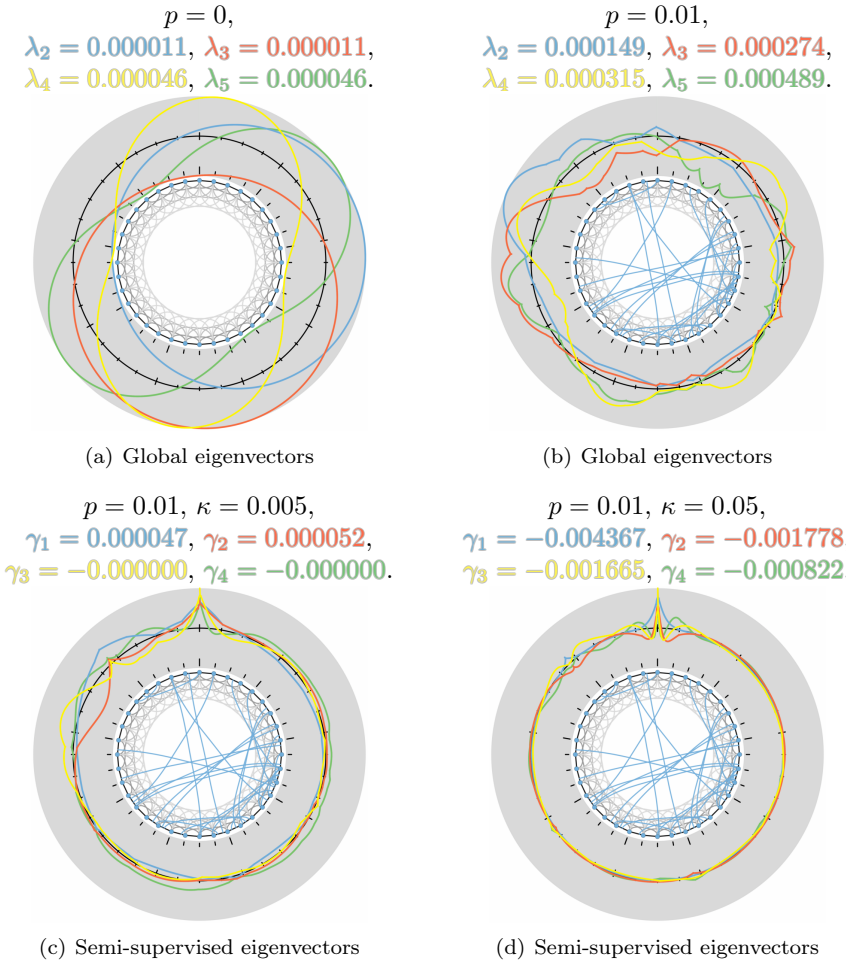
If we think about γ_k as being distinct eigenvalues of the generalized eigenvalue problem $\boldsymbol{L}_G\boldsymbol{x}_k = \gamma_k\boldsymbol{D}_G\boldsymbol{x}_k$, then it is clear that Eq 4.25, correctly computes the sequence of generalized eigenvectors. This is explained by the previously mentioned relationship with the Rayleigh quotient iteration, since the operator $\left(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{D}_G\right)$ will be superfluous as the global eigenvectors are already orthogonal in the degree-weighted norm.

4.2.2 Experiments on Small-world Networks

Finally, to demonstrate how semi-supervised eigenvectors can focus on specific target regions of a data graph and capture the slowest modes of local variation, we consider one-dimensional examples of the popular "small world" model [Watts and Strogatz, 1998]. This is a parameterized family of models that interpolates between low-dimensional grids and random graphs; and, as such, it allows us to illustrate the behavior of the method eigenvectors and its various parameters in a controlled setting, see Figure 4.3.

In Figure 4.3(a), we show a graph with no randomly-rewired edges ($p = 0$) and a locality parameter κ such that the global eigenvectors are obtained. This yields a symmetric graph with eigenvectors corresponding to orthogonal sinusoids, *i.e.*, for all eigenvectors, except the all-ones with eigenvalue 0, the algebraic multiplicity is 2, *i.e.*, the first two capture the slowest mode of variation and correspond to a sine and cosine with equal random phase-shift (rotational ambiguity). In Figure 4.3(b), random edges have been added with probability $p = 0.01$ and the locality parameter κ is still chosen such that the global eigenvectors of the rewired graph are obtained. In particular, note small kinks in the eigenvectors at the location of the randomly added edges. Since the graph is no longer symmetric, all of the visualized eigenvectors have algebraic multiplicity 1. Moreover, note that the slow mode of variation in the interval on the top left; a normalized-cut based on the leading global eigenvector would extract this region since the remainder of the ring is more well-connected due to the degree of rewiring. In Figure 4.3(c), we see the same graph construction as in Figure 4.3(b), except that the semi-supervised eigenvectors have a seed node at the top of the circle and the correlation parameter $\kappa_t = 0.005$. Note that, like the global eigenvectors, the local approach produces modes of increasing variation. In addition, note that the neighborhood around "11 o-clock" contains more mass, when compared with Figure 4.3(b); the reason for this is that this region is well-connected with the seed via a randomly added edge. Above the visualization we also show the γ_t that saturates κ_t; remember γ_t is the Lagrange multiplier that defines the effective correlation κ_t. Not shown is that if we kept reducing κ, then γ_t would tend towards λ_{t+1}, and the respective semi-supervised eigenvector would tend towards the global eigenvector. Finally, in Figure 4.3(d), the desired correlation is increased to $\kappa = 0.05$ (thus decreasing the value of γ_t), making the different modes of variation more localized in the neighborhood of the seed. It should be clear that, in addition to being determined by the locality parameter, we can think of γ as a regularizer biasing the global eigenvectors towards the region near the seed set.

Of course there are many other interesting aspects related to semi-supervised eigenvectors that could be covered here, such as the application of the Nyström



(a) Global eigenvectors

(b) Global eigenvectors

(c) Semi-supervised eigenvectors

(d) Semi-supervised eigenvectors

Figure 4.3: Illustration of small-world graphs with rewiring probability of $p = 0$ or $p = 0.01$. For each subfigure, the data consist of 3600 nodes, each connected to it's 8 nearest-neighbors and with different values of the κ parameter. In the center of each subfigure, we show the nodes (blue) and edges (black and light gray are the local edges, and blue are the randomly-rewired edges). We wrap around the plots (black x-axis and gray background), visualizing the 4 smallest non-trivial eigenvectors of the corresponding graph Laplacian. The eigenvectors are color coded as **blue**, **red**, **yellow**, and **green**, starting with the one having the smallest eigenvalue.

approximation covered in Section 3.3. However, as this chapter only serves as a compressed introduction to the field, I refer to [Hansen and Mahoney, 2013] that can be found in Appendix F, for more theoretical as well as empirical results with respect to semi-supervised eigenvectors.

Chapter 5

Neuroimaging

This chapter briefly introduces neuroimaging based on fMRI together with machine learning advances related to this particular field. Specifically we introduce [Hansen et al., 2012] that investigates a semi-supervised technique for modeling of the human decision process in real-time fMRI, and we also discuss the application of semi-supervised eigenvectors [Hansen and Mahoney, 2013] for data-driven feature extraction.

The Blood Oxygen Level Dependent (BOLD) signal is the most commonly used contrast in fMRI, primarily due to the relatively low complexity in terms of measuring, and its high Signal-to-Noise Ratio (SNR). Measurements of the BOLD signal is typically based on Echo Planar Imaging (EPI) techniques, where the actual measured effect is a result of signal loss due to the presence of deoxygenated haemoglobin, giving rise to a field inhomogeneity [Ogawa et al., 1992].

Traditionally, fMRI analyses have focused on characterizing the relations between cognitive variables and individual brain voxels, using *e.g.*, the mass-univariate General Linear Model (GLM), where statistical parametric maps are used to identify regions of gray matter that are significantly related to particular effects under study [Friston et al., 1994]. Even though the univariate approach have been tremendously productive, there are obvious limits on what can be learnt about cognitive states by only examining isolated voxels [Norman et al., 2006b].

Multivariate methods have therefore paved the way for more advanced paradigms involving complex cognition, where the latent brain state cannot solely be determined from looking at individual voxel time series [Bode and Haynes, 2009; Eger et al., 2008; Kamitani and Tong, 2005]. However, an immediate challenge for multivariate approaches are that weak signals carried by a sparse set of voxels can be very hard to detect, and for this reason multivariate approaches are often accompanied by spatial priors, to improve on the SNR.

5.1 Semi-supervised Learning for real-time fMRI

In a semi-supervised learning setting, only few training samples have an associated label. Such learning settings are very common in practice, as labeled data is expensive to produce and most often requires human interaction. We can think of a semi-supervised classifier as a generative model $p(\boldsymbol{x}, y)$ compared to a discriminative model $p(y|\boldsymbol{x})$ since the latter is independent of the input distribution $p(\boldsymbol{x})$. The intuition behind semi-supervised learning is therefore that better predictions can be made by learning the structure in the input distribution. There are many different branches of semi-supervised learning, *e.g.*, Expectation Maximization (EM) with generative mixture models, self-training, co-training, transductive SVMs, and graph-based methods [Zhu, 2005]. In particular, this section introduces [Hansen et al., 2012], found in Appendix C, that considers a semi-supervised modeling technique that uses the graph Laplacian as a regularizer, exploiting the smoothness of fMRI data.

In a real-time Brain-Computer Interface (BCI) fMRI setup, such as the one

visualized in Figure 5.1, the signal processing pipeline can be very flexible, and we are not tied to a usual block design. However, this flexibility can make it difficult to obtain *true* labels, *i.e.*, it may be difficult to quantify the exact brain state of the subject at a given point in time, and so we face the risk of "confusing" the downstream classification algorithm.

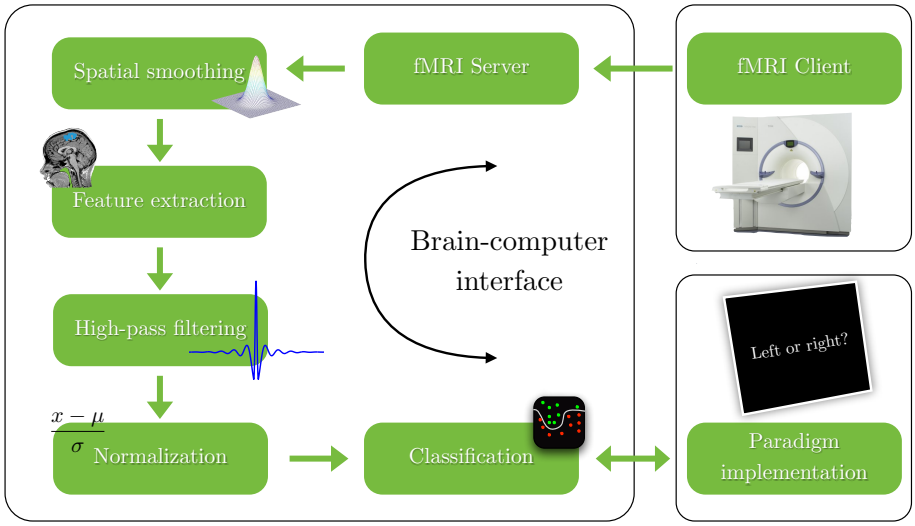


Figure 5.1: Shows the component organization of our BCI pipeline. Volumes are streamed over a network connection to a computer responsible for various usual preprocessing stages, where arrows indicate the direction of the flow. The classification stage receives preprocessed fMRI volumes and communicates with the paradigm implementation, by sending predictions and receiving subject events in the form of button presses.

We address mislabeling issues in paradigms involving complex cognition, by considering a manifold regularizing prior for modeling a sequence of neural events leading up to a decision. The method can explain nonlinear relations, it is directly applicable for online learning in the context of real-time fMRI, and our experimental results show that the method can efficiently avoid model degeneracy caused by mislabeling [Hansen et al., 2012]. Compared to linear approaches, nonlinear studies suggest that complex interactions in brain patterns are important for distinguishing cognitive states [Davatzikos et al., 2005; Haynes et al., 2007; Kriegeskorte et al., 2006; LaConte et al., 2007], and recent experimental results indicate that the manifold assumption is valid for fMRI data [Blaschko et al., 2009].

To model the intrinsic decision process more naturally we impose a manifold

regularizing prior, and thereby rely on the smoothness of fMRI data to infer the brain state of each individual sample. Specifically, using the normalized graph Laplacian we define a smoothness operator that takes the unlabeled data into account, *i.e.*, we seek functions $\boldsymbol{f}$ that agree with the labeled data, but also smooth with respect to the graph. The smoothness measured by the graph Laplacian is given by $\boldsymbol{f}^\top \mathcal{L}_G \boldsymbol{f}$ that we can view as a zero mean Gaussian process $\boldsymbol{f} \sim \mathcal{N}(0, \mathcal{L}_G^{-1})$.

To aggregate a trial decision $\boldsymbol{y}$ we consider the process $\boldsymbol{x} \rightarrow \boldsymbol{f} \rightarrow \boldsymbol{z} \rightarrow \boldsymbol{y}$. Stated in words; the smoothness of $\boldsymbol{x}$ measured by $\boldsymbol{f}$ is mapped to a prediction $\boldsymbol{z}$ per sampled fMRI volume; then all predictions for the current trial are aggregated into a single decision $\boldsymbol{y}$, *i.e.*, a probability distribution over possible conditions. Hence, we seek a good point estimate of the joint posterior

$$p(\boldsymbol{f}_{i\in\mathcal{K}}, \boldsymbol{\theta}|\boldsymbol{y}_k) \propto \int p(\boldsymbol{y}_k|\boldsymbol{\theta}, \boldsymbol{z}_{i\in\mathcal{K}}) p(\boldsymbol{z}_{i\in\mathcal{K}}|\boldsymbol{f}_{i\in\mathcal{K}}) p(\boldsymbol{f}_{i\in\mathcal{K}}) p(\boldsymbol{\theta}) \partial \boldsymbol{z}_{i\in\mathcal{K}}, \tag{5.1}$$

where $\mathcal{K}$ is the set of volume indices associated with the k^{th} trial, and $\boldsymbol{\theta}$ is a model parameter. We model a C class classification problem by considering a likelihood based on a Softmax Function Model (SFM)

$$p(\boldsymbol{z}_i|\boldsymbol{f}_i) = \prod_{j=1}^{C} \left(\frac{\exp(\boldsymbol{f}_i^j)}{\sum_{k=1}^{C} \exp(\boldsymbol{f}_i^k)} \right)^{\boldsymbol{z}_i^j}, \tag{5.2}$$

where $\boldsymbol{f}_i = f(\boldsymbol{x}_i)$ are the latent variables generated by the process $\boldsymbol{x} \rightarrow \boldsymbol{f} \rightarrow \boldsymbol{z}$, and $\boldsymbol{z}_i = [0, \ldots, 0, \boldsymbol{z}_i^j = 1, 0, \ldots, 0]$ encodes that $\boldsymbol{x}_i$ belongs to class j ($j = 1, 2 \ldots, C$). Moreover, we use $\boldsymbol{z}_i = \boldsymbol{0}$ to denote an unlabeled sample. See [Hansen et al., 2012] in Appendix C for further details regarding the model and the respective inference procedure.

We tested the method on BOLD sensitive fMRI data acquired on a 3 Tesla MR scanner (Siemens Magnetom Trio). During the scanning session (800 volumes) the subject was engaged in a simple motor paradigm in which the subject was asked to respond by either left or right index finger keypress when a visual cue was presented. The model was used to predict which finger (left or right) the subject selected to press the button with. Pre-processing steps included: rigid body realignment, spatial smoothing (6 mm full width at half maximum isotropic Gaussian kernel), high-pass filtering (cut-off frequency 1/128 Hz), and static masking of premotor cortex. In terms of our approach, we consider a 3 class classification problem, classifying between *baseline*, *left*, and *right*, and we validated the approach against the popular SVM objective, that has proven to yield good generalization performance in a variety of fMRI studies, see for example [LaConte et al., 2007]. In terms of the suggested modeling approach we define the graph using the Gaussian kernel $w_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$, which was also used in the SVM.

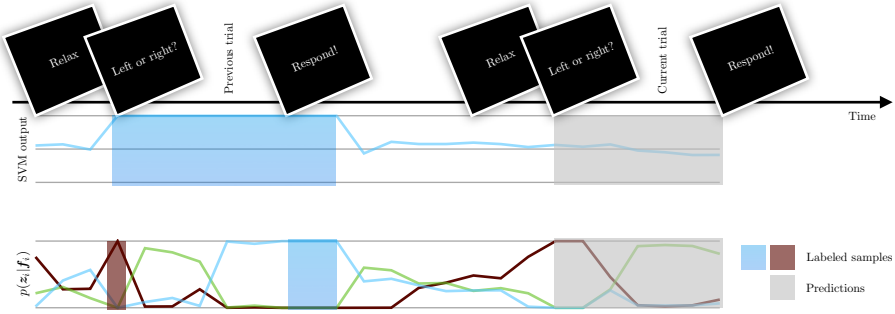


Figure 5.2: Illustrates a conceptual overview (best viewed in color) of the training process of our approach and the SVM used for comparison. The SVM (upper plot) is trained on both the preparation and response samples (highlighted in blue), whereas remaining samples correspond to predictions, *i.e.*, the SVM is trained using samples in this interval through all preceding trials. The semi-supervised approach (lower plot) is trained on a few baseline samples (highlighted in red) and 3 samples around the end of each trial (highlighted in blue), *i.e.*, the three time series correspond to the probability of baseline (red), left (blue) and right (green). Common for both approaches; highlighted in grey are the preparation predictions that must be aggregated into the decision of the current trial, before the subject reveals the actual decision. The classification accuracy is measured as the number of correctly aggregated trial decisions.

Figure 5.2 illustrates a conceptual overview of the training approach. For both the proposed modeling technique, and the SVM used for comparison, we train another kernelized SVM on the predictions of each respective model, to aggregate predictions of a trial $\boldsymbol{z}$ to a decision $\boldsymbol{y}$ in a systematic manner. We analyzed a single 44 trial scanning session, and measured performance on the final 29 trials, *i.e.*, as we learn and predict in an online fashion we let both approaches stabilize using the leading 15 trials.

The suggested method reached an accuracy of ≈ 0.76, whereas the SVM achieved ≈ 0.73, *i.e.*, we need more data to state significant performance improvements, but we do see indications supporting the suggested modeling approach. One reason why the SVM performs relatively well on the ambiguous data may be explained through slackness regularization, but from a modeling perspective the approach is less attractive since mislabeled samples are then treated as outliers, hence, in the SVM mislabeled samples will become support vectors.

The suggested lazy labeling scheme makes few assumptions about the temporal dynamics of the brain state by only assigning hard labels to a few volumes within each block, while still benefitting from unlabeled samples by identifying the manifold on which the data reside. In essence the suggested classification scheme allows the temporal dynamics of a decision process to be captured by modeling the identification sequence of related neural events leading to a decision. The results presented in [Hansen et al., 2012] indicate that the suggested labeling scheme performs on par with existing state of the art nonlinear methods, and on synthetic data we show significant improvements over classical modeling techniques used in fMRI.

5.2 Semi-supervised Eigenvectors for fMRI

Unsupervised data-driven dimensionality reduction techniques are widely used in modalities such as fMRI and ElectroEncephaloGraphy (EEG). In fMRI, activation sources are typically assumed to be linearly mixed, as assumed by PCA and Independent Component Analysis (ICA). However, multiple studies indicate that event-related as well as resting state fMRI exhibit nonlinear properties, *e.g.*, artifacts caused by subject movements [Lund et al., 2005], and in such cases the aforementioned approaches may prove insufficient for modeling. Despite of these shortcomings PCA has successfully been applied in multiple studies [Friston, 1998; Moeller and Strother, 1991], and so has ICA [Beckmann et al., 2005; Calhoun et al., 2001; McKeown et al., 1997].

Among nonlinear dimensionally reduction techniques are for instance Local Linear Embedding (LLE), a local eigenvector-based approach to find the low-dimensional embedding of data points, such that each point in the dataset can be described by a linear combination of its nearest neighbors. In terms of fMRI analysis, LLE has successfully been used to model resting-state networks, but also as a nonlinear preprocessing step for ICA [Mannfolk et al., 2010]. Moreover, nonlinear relations were also discovered in spatiotemporal fMRI studies [Lahaye et al., 2003; Xie et al., 2008] where nonlinear modeling approaches resulted in the best model for functional connectivity. Along the lines of functional connectivity, clustering algorithms such as K-Means (KM) provide a simple and fast approach, that unfortunately may prove too simplistic as each time series is assumed to be drawn from one of k isotropic gaussians [Venkataraman et al., 2009]. Functional connectivity has also been well-modeled by graph based techniques such as spectral clustering [Venkataraman et al., 2009] and normalized cuts [van den Heuvel et al., 2008]. For further details, I refer to [van den Heuvel and Hulshoff Pol, 2010], who provides a broad overview of connectivity analyses.

Searchlight is an algorithm that scans through the whole brain by running multiple multivariate Region Of Interest (ROI) analyses, measuring the respective generalization performance, and outputs a brain map showing which regions exhibit the best discriminative properties, for example measured by classification accuracy for a particular subject task [Kriegeskorte et al., 2006]. This approach was for instance applied by [Haynes et al., 2007], who used it to find regions in the brain that are predictive with respect to human intentions. Compared to a univariate approach, searchlight takes advantage of multivariate techniques, with the caveat that it only performs well if the target signal is available within the area covered by the ROI. This limitation is indeed shared by the univariate approaches, but with searchlight we have the freedom to increase or decrease the ROI, depending on the structure of the considered problem. If the ROI is small we approach a univariate analysis, whereas if the ROI is large, the information localization becomes less specific. Thus, if the multivariate signal is spatially distributed the searchlight approach will fall short, and simply increasing the ROI may not be a solution as irrelevant time series will decrease the SNR.

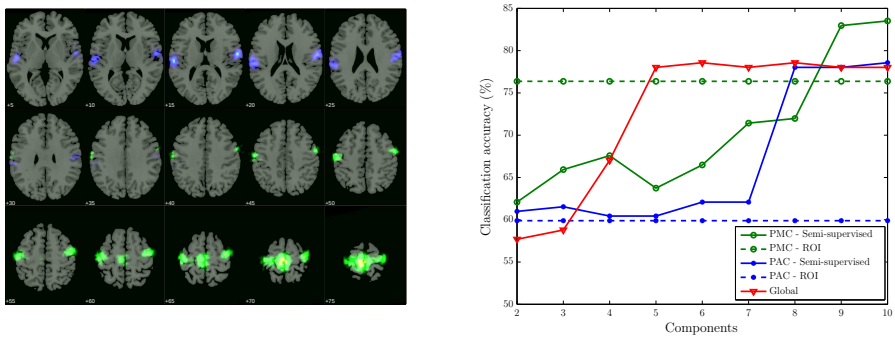
The semi-supervised eigenvectors introduced in Section 4.2 can be used to construct a spatially guided basis, that naturally allows for spatially distributed signal representations. This strategy shares many similarities with the searchlight approach, but is not tied to a particular ROI, and can span distributed voxel time series that are similar in terms of our graph representation. Using the semi-supervised eigenvectors on the voxel $\times$ voxel similarity graph in this way, will yield a low dimensional representation that we can project the fMRI voxel time series onto and in that projected space we can apply any suitable classification algorithm.

Here we demonstrate a few results based on BOLD sensitive fMRI data acquired on a 3 Tesla MR scanner (Siemens Magnetom Verio). Additional sequence parameter were as follows: 25 interleaved echo planar imaging gradient echo slices, echo time 30 ms, repetition time 1390 ms, flip angle 90 degrees. During the scanning session (1300 volumes) the subject was engaged in a simple motor paradigm in which the subject was asked to respond with key-presses when a visual cue was presented, and the classification task is to detect such key-presses. Pre-processing steps included: rigid body realignment, spatial smoothing (6 mm full width at half maximum isotropic Gaussian kernel), and high-pass filtering (cut-off frequency 1/128 Hz).

We construct a voxel $\times$ voxel 10-nearest neighbor graph using the nonlinear affinity $w_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$. Using a probabilistic atlas created by averaging across multiple subjects [Eickhoff et al., 2005], we carry out two experiments based on semi-supervised eigenvectors. Specifically, we construct semi-supervised eigenvectors seeded in Primary Motor Cortex (PMC), known to be highly involved in the subject task [Geyer et al., 1996], as well as semi-

supervised eigenvectors seeded in Primary Auditory Cortex (PAC), that is not expected to carry much signal with respect to our target variable [Morosan et al., 2001]. Both seed regions are highlighted in Figure 5.3(a).

For comparison we consider the leading global eigenvectors of the graph Laplacian, as well as simply extracting the time series as specified by the seed regions. For all of the considered feature extraction approaches we feed either the projected or extracted time series into a linear SVM responsible for the downstream classification task.



(a) Seed regions

(b) Classification accuracy

Figure 5.3: In Figure 5.3(a) we show the two considered seed regions; blue corresponds to PAC whereas green to PMC. The plot in Figure 5.3(b) shows the classification accuracy for the 5 different features extraction approaches. The dashed lines mark the reference where all voxel time series, as covered by the seed, are used in the downstream classifier, whereas the solid ones correspond to the accuracy obtained from projecting the data onto the semi-supervised eigenvectors seeded in PAC and PMC, as well as the global eigenvectors.

Figure 5.3(b) summarizes the classification accuracies obtained by performing leave-one-out cross validation as a function of the number of components, and for each semi-supervised eigenvector we fix $\kappa = \frac{1}{k}$ where k is the number of components. Hence, for two components, each correlates 0.5 with the seed, and so forth. In the same plot, the dashed blue lines corresponds to classifying the brain state using only voxel time series in the region as defined by PAC. Unsurprisingly, for the dashed green line, corresponding to PMC, it is evident that the primary motor cortex is a much better proxy for predicting motor responses. Due to inter-subject variability there is no guarantee that the rigid body realignment will align the seed perfectly with the physical region, which

explains why the data-driven global eigenvectors are able to yield an even higher accuracy than the PAC time series. Also seen is the "bump" in classification accuracy for the global eigenvectors, when we reach 4-5 components. Thus, for this particular dataset, relevant parts of the are signal are captured in this regime.

For few components the semi-supervised approaches are too localized to explain relevant local heterogeneities both near and within the seed set. As we increase the number of components they become less localized, and the semi-supervised eigenvectors seeded in PMC eventually surpasses the accuracy of global approach. As we consider more and more components, while distributing the correlation even across the semi-supervised eigenvectors, they will eventually converge to the global eigenvectors. Finally, in the limit of a single component, the projection onto the leading trivial global eigenvector will simply correspond to the average time series, whereas for a leading semi-supervised eigenvector the solution is simply the seed itself, *i.e.*, the projection onto this corresponds to a weighted average in the region defined by the seed. Hence, as seen in Figure 5.3(b) there exists a regime in which the semi-supervised approach performs better as we are able to pickup the relevant local heterogeneities at that particular scale, given that the seed is relevant with respect to the subject task. See [Hansen and Mahoney, 2013] for further details regarding this analysis.

Chapter 6

Topic Modeling

This chapter provides a brief introduction to topic modeling, and considers the GPU accelerated framework in [Hansen et al., 2011b] that was used for a large-scale implementation of the IRM.

Mining for associations in large-scale graphs is a problem of both theoretical and practical importance, *e.g.*, study of social networks, collaborative filtering, and in web scale text processing. Hence, such problems mount the need for computationally efficient tools that can accurately predict relations and also provide relevant insights in the underlying mechanisms. In topic modeling, co-clustering refers to the simultaneous clustering of rows and columns in a matrix. Co-clustering has gained popularity in several fields, including bio-informatics for identification of co-regulated genes and gene functional annotation [Madeira and Oliveira, 2004], collaborative filtering and market basket analysis for identification of user and product segments [Wang et al., 2002], text miming [Berkhin and Becher, 2002; Dhillon, 2001; Xu et al., 2006] for identification of related terms and documents, and social network modeling to find relationships between agents and behavior [Barber et al., 2008; Fortunato, 2010]. A plethora of co-clustering methods have been proposed over the years including the iterative refinement approach of [Hartigan, 1972], heuristics for grouping columns and rows in an adjacency matrix [Tanay et al., 2004], spectral [Dhillon, 2001] and information theoretic approaches [Dhillon et al., 2003], and methods inspired by community detection in complex networks [Fortunato, 2010; Reichardt and Bornholdt, 2007; Xu et al., 2006]. For reviews on co-clustering see [Fortunato, 2010; Madeira and Oliveira, 2004; Mechelen et al., 2004].

Co-clustering can be defined as the matrix decomposition problem

$$\boldsymbol{A} \approx \boldsymbol{R} = \boldsymbol{Z}^{(1)^\top} \boldsymbol{\eta} \boldsymbol{Z}^{(2)},$$

where the data matrix $\boldsymbol{A}$ is decomposed into clusters, grouping the row and column entries of the matrix, respectively in $\boldsymbol{Z}^{(1)^\top}$ and $\boldsymbol{Z}^{(1)^\top}$ such that the data matrix is segmented into homogenous regions specified by the matrix $\boldsymbol{\eta}$. For an illustration of the co-clustering problem, see also Figure 6.1 where the re-permutation by $\boldsymbol{P}$ of $\boldsymbol{A}$ reveals the block structure identified by the latent variables.

In [Hansen et al., 2011b] we exploit that the alternating estimation of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ while keeping the remaining model parameters fixed, parallelizes over the rows and columns respectively of the data matrix. Specifically, we exploit this particular structure to carry out efficient inference on GPUs, that previously have been explored in [Liu et al., 2010; Papadimitriou and Sun, 2008] for MapReduce and similar large-scale cloud environments [Ramanathan et al., 2010]. Moreover, we approach the co-clustering problem in terms of generative models that serve two main purposes in machine learning, namely to provide interpretation and domain insight via the inference of relevant latent variables, as well as identify predictive relations. In particular, we make use of non-parametric learning that admits to infer the number of row and columns clusters from a hypothesis space of potentially infinitely many clusters [Antoniak, 1974;

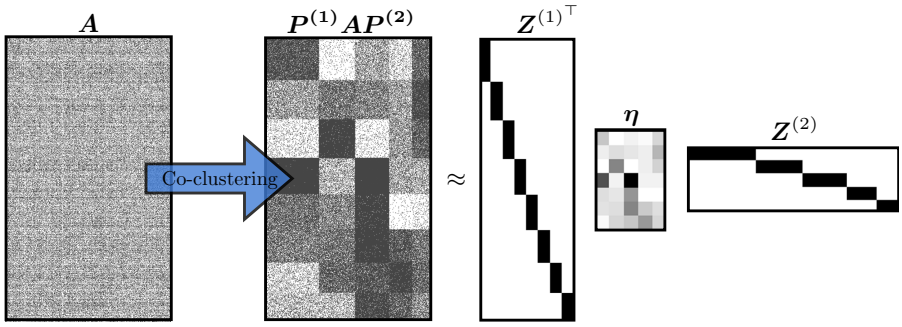


Figure 6.1: Illustration of co-clustering. The permuted data matrix $\boldsymbol{P^{(1)}AP^{(2)}}$ is decomposed into row and column clusters $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ such that η_{lm} gives the extend in which row group l relates to column group m. As such, co-clustering segments the data matrix $\boldsymbol{A}$ into homogenous blocks.

Kemp et al., 2006; Pitman, 2002], as previously considered for the Bernoulli likelihood in [Kemp et al., 2006; Xu et al., 2006] forming the IRM.

The IRM can be cast as co-clustering approach for bipartite networks where the nodes of each mode are grouped simultaneously. A benefit of the IRM over existing co-clustering approaches is that the model explicitly exploits the statistical properties of binary graphs and allows the number of components of each mode to be inferred from the data. Specifically, in [Hansen et al., 2011b] we reach web scale applications with the bipartite IRM model by exploiting the previously mentioned independence between $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$. This GPU implementation achieves a speedup of two orders of magnitude for model estimation on large networks ($N = 10^7 - 10^9$) compared to estimation based on conventional CPUs. Moreover, with this implements we analyze networks with more than 100 million links on which model estimation can be done within an hour on a single GPU.

To motivate relational models in general as well as our large-scale implementation of the IRM, the following shows examples of a few topic groups extracted from abstracts of U.S. National Library of Medicine (PubMed) based on the bag of words representation, publicly available from [Frank and Asuncion, 2010]. This particular dataset contains roughly 140K words and 8 million documents with approximately 500 million word occurrences:

Neurology group; *amygdala, ca1, ca3, caudate, cell bodies, circuitry, colliculus, dendrites, dentate gyrus, dorsal root, dorsolateral, entorhinal, gabaergic, ganglion cell, geniculate, innervated, interneuron, lesioned, limbic, mesencephalic,*

midbrain, motoneuron, myelinated, neocortex, neocortical, nerve fiber, neuropil, perikarya, purkinje, putamen, raphe, rostral, soma, somata, substantia nigra, synapse, tegmental, thalamic, thalamus.

Cancer group; *adenocarcinoma, adjuvant, benign, biopsies, biopsy, breast, breast cancer, cancer patient, carcinoma, carcinomas, cell carcinoma, chemotherapy, grade,histologic, histological, histologically, histology, hyperplasia, invasion, invasive, irradiation,lung cancer, lymph node, lymph nodes, malignancies, malignancy, malignant, metastases, metastasis, metastatic, neoplasm, neoplastic, prognostic, prostate, radiation, radiotherapy, recurrence, tumour.*

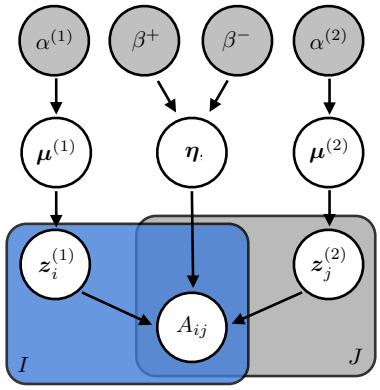
Cardiology group; *angina, antiarrhythmic, arrhythmia, arrhythmias, atrial, atrial fibrillation, atrioventricular, atrium, av, beat, cardiomyopathy, congestive heart, ecg, echocardiographic, ejection fraction, electrocardiogram, electrocardiographic, electrophysiologic, end-diastolic, endocardial, epicardial, fibrillation, implantable, left atrial, left ventricle, lv, myocardial ischemia, pacemaker, pacing, pectoris, qr, right ventricular, tachycardia,ventricular function, ventricular hypertrophy, ventricular tachycardia.*

The remainder of this chapter briefly introduces the GPU implementation of the IRM. For further details, specifically regarding other types of relational models, I refer to the full paper [Hansen et al., 2011b] that can be found in Appendix A.

6.1 The Infinite Relational Model

The graphical model for co-clustering based on the Bernoulli likelihood for binary bipartite graphs is visualized in Figure 6.2.

For each mode, a node is assigned to a cluster by drawing from a discrete distribution parameterized by $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ respectively. $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ are in turn drawn from a Dirichlet distribution parameterized by $\alpha^{(1)}/K^{(1)}\mathbf{1}$ and $\alpha^{(2)}/K^{(2)}\mathbf{1}$. Then, the entry in the matrix $\boldsymbol{A}_{ij}$ is according to the co-clustering model governed by the class assignment of the i^{th} row and j^{th} column, *i.e.*, $\boldsymbol{z}_i^{(1)}$ and $\boldsymbol{z}_j^{(2)}$ as well as the inter-group relations $\boldsymbol{\eta}$ where η_{lm} gives the probability of observing a link between class l and m for the Bernoulli likelihood. Entries in the data matrix $\boldsymbol{A}$ are modeled conditionally independent given their respective row and column assignments. Finally, to account for an infinite number of clusters we can take the limits $K^{(1)} \to \infty$ and $K^{(2)} \to \infty$ which has an analytic solution formed by the Chinese Restaurant Process (CRP) [Antoniak, 1974; Kemp et al., 2006; Pitman, 2002], *i.e.*, in this limit the Dirichlet distribution converges to the CRP.



The Bipartite Infinite Relational Model

$$\begin{aligned}
\boldsymbol{\mu}^{(1)} &\sim \text{Dirichlet}(\alpha^{(1)}/K^{(1)}\mathbf{1}) \\
\boldsymbol{\mu}^{(2)} &\sim \text{Dirichlet}(\alpha^{(2)}/K^{(2)}\mathbf{1}) \\
\eta_{lm} &\sim \text{Beta}(\beta^+, \beta^-) \\
\boldsymbol{z}_i^{(1)} &\sim \text{Discrete}(\boldsymbol{\mu}^{(1)}) \\
\boldsymbol{z}_j^{(2)} &\sim \text{Discrete}(\boldsymbol{\mu}^{(2)}) \\
A_{ij} &\sim \text{Bernoulli}({\boldsymbol{z}_i^{(1)}}^\top \boldsymbol{\eta} \boldsymbol{z}_j^{(2)})
\end{aligned}$$

Figure 6.2: A graphical representation of the IRM for bipartite graphs.

As the Dirichlet distribution is conjugate to the Discrete distribution and the priors on $\boldsymbol{\eta}$ are conjugate, we can analytically integrate over $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}$ and $\boldsymbol{\eta}$ to collapse the model. However, the marginalization over these variables introduce a dependence within each mode such that the assignments of each row in mode one, and each column in mode two no longer can be updated in parallel. Hence, rather than collapsing the parameters of the model, we use blocked Gibbs sampling which allows for efficient parallelization and that potentially also result in better mixing [Ishwaran and James, 2001; Z. Xu, 2007].

The blocked Gibbs sampling approach leads to the following parameter updates for $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$

$$P\left(z_i^{(1)} = l | \boldsymbol{A}, \boldsymbol{Z}^{(2)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\eta}\right) \propto \mu_l^{(1)} \exp\left(\sum_{m,j} \log\left(\tfrac{\eta_{lm}}{1-\eta_{lm}}\right) A_{ij} z_{mj}^{(2)} + \log(1-\eta_{lm}) z_{mj}^{(2)}\right), \tag{6.1}$$

and

$$P\left(z_j^{(2)} = m | \boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}\right) \propto \mu_m^{(2)} \exp\left(\sum_{l,i} \log\left(\tfrac{\eta_{lm}}{1-\eta_{lm}}\right) A_{ij} z_{li}^{(1)} + \log(1-\eta_{lm}) z_{li}^{(1)}\right). \tag{6.2}$$

Hence, all $z_i^{(1)}$ can be sampled in parallel given $\boldsymbol{Z}^{(2)}$, and vice versa for $z_j^{(2)}$ given $\boldsymbol{Z}^{(1)}$. As the remaining update rules are irrelevant with respect to the GPU implementation that will be described in the following section, I refer

to [Hansen et al., 2011b] in Appendix A for further details regarding blocked Gibbs sampling.

6.2 GPU Implementation

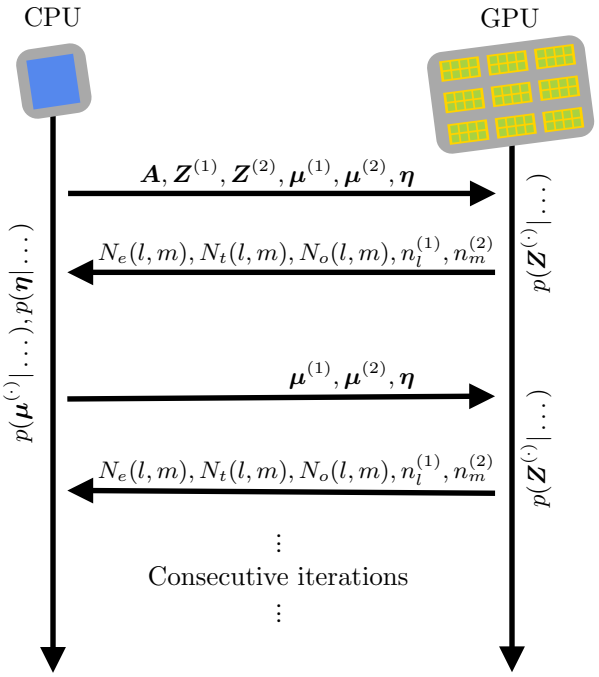


Figure 6.3: A simplified view of how model inference is distributed between the CPU and GPU. Initially (top) $\boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}$ are uploaded to the GPU, assuming a sufficient amount of memory. In each iteration, we sample the distributions $p(\boldsymbol{Z}^{(1)}|\dots) \nparallel p(\boldsymbol{Z}^{(2)}|\dots)$, compute the sufficient statistics, $N_e(l,m), N_t(l,m), N_o(l,m), n_l^{(1)}, n_m^{(2)}$, and download these to the CPU code. The sufficient statistics are used when sampling $p(\boldsymbol{\mu}^{(1)}|\dots) \parallel p(\boldsymbol{\mu}^{(2)}|\dots) \parallel p(\boldsymbol{\eta}|\dots)$, see [Hansen et al., 2011b]. In case of sufficient GPU memory, consecutive sampling iterations only upload $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}$, otherwise $\boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}$ are decomposed into suitable sized blocks that are also uploaded and processed in turn.

In recent years, the GPU has evolved into a highly parallel, multithreaded, many-core processor that is not only specialized for graphics rendering, but also for generic intensive parallel computations. Compared to a CPU which is well-suited for processing code with a complex control flow, a GPU is much better suited for addressing problems that can be expressed as data-parallel computations with a high arithmetic intensity. For our implementation we used CUDA that is a set of compiler tools and language extensions developed by NVIDIA, and that enable programmers to code algorithms for execution on the GPU. At its core, CUDA uses three key abstractions: thread groups; shared memories; and barrier synchronizations. These are exposed to the programmer as a minimal set of extensions to C/C++. A CUDA-capable GPU consists of a set of Multi Processors (MP), each containing multiple Scalar Processors (SP), as well as different types of local memories that the SPs may access. All MPs have also access to a large global memory that, compared to their internal memories, is much slower to access. A computation task to be executed on a CUDA-capable device, is setup in a grid, where each element in the grid gets assigned to a thread. The grid is then decomposed into blocks that are scheduled onto the MPs with available resources, and the assigned MP will schedule the elements of the block onto its SPs in warps with 32 threads. The best performance is obtained when all threads in a warp execute the same instruction and when the total number of threads in the grid is large. This allows for some of the various overhead latencies to be overlapped with arithmetic operations.

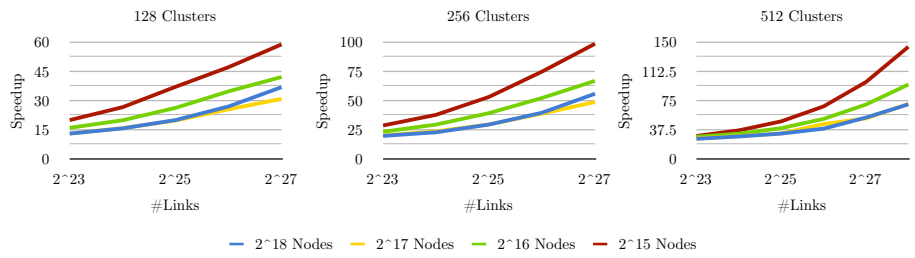


Figure 6.4: The speedup in time for GPU computing compared to computing on the CPU when varying the number of links and nodes of the graph. The number of nodes for the two modes are here identical. Left panel; speedup for 128 clusters in each mode, middle panel; speedup for 256 clusters in each mode, right panel; speedup for 512 clusters in each mode.

The implementation in [Hansen et al., 2011b], consists of a CPU part for sampling $\boldsymbol{\eta}$, $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}$ and a GPU part for sampling $\boldsymbol{Z}^{(1)}$, $\boldsymbol{Z}^{(2)}$. The main reason for not keeping everything in GPU code is that only the sampling of the cluster assignment matrices exhibit an arithmetic intensity sufficient for taking advantage

of the GPU architecture, see also Figure 6.3 that illustrates how the computations are divided between the CPU and GPU. One of the greatest challenges in the development of high performance GPU applications is to keep device memory latency low and thereby all threads occupied with arithmetic operations. Current architectures allow memory transfers to be grouped into so-called coalesced memory transfers when all addresses of an executing warp falls within a single memory segment [NVIDIA, 2008]. Unfortunately, sparse matrix representations often result in scattered memory accesses thereby making efficient memory segmentation difficult. Performing sparse matrix operations on CUDA capable GPUs have already been analyzed [Bell and Garland, 2008], however as the posterior likelihood of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ contain structure that can be exploited to yield a more efficient memory pattern, no existing implementation seems ideal for our purpose. To minimize the memory footprint of the cluster assignment matrices ($\boldsymbol{Z}^{(1)}$, $\boldsymbol{Z}^{(2)}$) we exploit the property that a node can only belong to a single cluster, hence they are encoded as vectors where each entry corresponds to the respective cluster number. With respect to the adjacency matrix $\boldsymbol{A}$, we presort each mode according to node degree and chunk it into aligned blocks of memory where each entry encodes the position of a link. Using the proposed format the calculation of $\boldsymbol{A}\boldsymbol{Z}^{(2)^\top}$ and $\boldsymbol{A}^\top\boldsymbol{Z}^{(1)^\top}$ (constituting the main computational bottleneck in the updates of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ respectively) corresponds to incrementing link counts in the dense result matrix directly indexed by our sparse representation. The implemented memory layout minimizes thread divergence and provides coalesced memory transfers in $\boldsymbol{A}$, whereas the scattered accesses in $\boldsymbol{Z}$ can be cached in local memory. Remaining dense matrix operations are handled using the CUBLAS library[1], whereas we utilize yet another custom CUDA kernel to sample the cluster assignment matrices done by applying inverse transform sampling requiring a stream of uniform random numbers per thread, that we also generate within GPU code, see [Nguyen, 2007].

Finally, in order to evaluate the speedup of GPU computation over traditional CPU computation we analyze synthetically generated graphs, varying the number of nodes, links and clusters. We highlight that the benchmark compares the GPU code with a semantically equivalent CPU implementation based on the Intel Math Kernel Library (MKL). Figure 6.4 shows the speedup which is positively correlated with the number of links and clusters, since increasing these will improve the overall arithmetic intensity on the GPU. Increasing in the number of nodes while keeping the number of links fixed yields a negative impact in the speedup, because increasing the sparsity reduces the load on each parallel processing GPU thread. In particular, for 512 clusters our implementation peaks with more than a 140 times speedup, and for further details regarding implementation and hardware setup I refer to [Hansen et al., 2011b] in Appendix A.

[1] A GPU accelerated version of BLAS developed NVIDIA

Chapter 7

Conclusions

This chapter concludes on each published and submitted contribution of this dissertation.

In [Hansen et al., 2011b] we developed an efficient framework for GPU accelerated model estimation for relational models. Using this efficient GPU implementation we demonstrated, to our knowledge, the first large-scale co-clustering results based on the IRM. In terms of scalability the current GPU implementation allows us to go beyond the datasets analyzed here and make inference in even larger bipartite networks, provided that the cluster assignment matrices can fit in device memory. Furthermore, the memory organization that is exploited by the implementation makes it straightforward to utilize systems with multiple GPUs, simply by parallel distribution of discretized subproblems, *i.e.*, a heterogeneous setup with different levels of parallelism. Moreover, we note that the GPU framework trivially generalizes to other inference approaches such as variational Bayes [Z. Xu, 2007]. Future work will focus on extending the proposed framework for graphs with temporal dynamics, *i.e.*, online auctions and collaborative content creation such as blogs. Finally, we imagine a potential in exploiting random projection methods to efficiently subsample data while still recovering network structure supported by the data within the available timeframe imposed by online systems.

In [Hansen et al., 2011b] we consider how maximizing the radius of the MEB of the cluster means in the RKHS provides a meaningful heuristic for finding the optimal hyperparameters in supervised kernel learning. Compared to other distance metrics in a RKHS we found that the MEB approach provides better results together with an attractive time complexity, achieved by exploiting the randomization technique of [Clarkson et al., 2010]. This makes the approach very useful for crude hyperparameter selection in large-scale multi-class problems, where CV proves computationally infeasible. Future research in this area includes testing on a wider range of large-scale problems, outlier detection measured in terms of the MEB, and generalize the technique to ellipsoids in order to accommodate for covariance structures.

In [Hansen et al., 2012] we exploited semi-supervised learning to relax the labeling scheme in fMRI data analysis. The scheme makes very few assumptions about the temporal dynamics of the brain state, in that only hard labels are given to a few volumes within each trial, while the techniqueue still benefits from unlabeled samples by identifying the manifold on which the data reside. In essence the suggested classification scheme will, to some extend, allow the non-stationary temporal dynamics of a decision process to be captured, enabling the identification of sequences of related neural events that ultimately leads to a decision. Results based on synthetic data demonstrate significant enhancements, while the results indicate improvements over existing state of the art nonlinear methods on real data. Future research in this area should focus on a more comprehensive evaluation on a wide range of fMRI datasets. Also, by analyzing the stationarity of the suggested model across subjects, we can investigate the generalizability of the human decision process.

In [Hansen and Mahoney, 2012] we introduced the notion of semi-supervised eigenvectors of a graph Laplacian, motivated by the previous work of [Mahoney et al., 2012]. This methodology was further generalized in [Hansen and Mahoney, 2013] where variants of the algorithm that generalize to large-scale kernel machines, and large-scale data graphs, were introduced. Specifically, for kernel machines, low-rank matrix decompositions have recently gained popularity in scaling up kernel methods to huge amounts of data, under the assumption that the kernel matrix used for encoding the similarity between data samples can be well-approximated by a few eigenvectors due to a rapidly decaying spectrum [Williams and Seeger, 2000]. In effect, we derived a solution for semi-supervised eigenvectors, based on the recently-popular Nyström approximation to speed up the computation, and we did so by considering how a low-rank decomposition can be exploited to yield solutions, where the running time largely depends on a matrix-vector product. Regarding large-scale graphs, we specifically focused on the Push algorithm by [Andersen et al., 2006], that approximates the solution to PageRank very efficiently, and we showed how multiple semi-supervised eigenvectors can be computed solely in terms of diffusion processes. This makes the algorithm very scalable and applicable for large-scale data, since only the local neighborhood near the seed set will be touched, as opposed to solving the linear system of equations that explicitly touches all nodes in the graph. Future research in this area, will involve finding more suitable applications to demonstrate advantages, both in terms of scalability as well as the ability to extract relevant information, as compared to the usual global eigenvectors. Especially, in fMRI data analysis the semi-supervised eigenvectors may prove very useful as an alternative to the Searchlight algorithm [Kriegeskorte et al., 2006], in that the semi-supervised eigenvectors are data-driven, and efficiently approximates the combinatorial NP-hard problem of finding a good subset of voxel that are related to a particular effect under study.

In [Hansen et al., 2013] we proposed two techniques for improving denoising by kernel PCA. We extended the work of [Walder et al., 2010a] to allow for more than one basis vector, leading to a more general semi-supervised kernel PCA approach that extends to a multidimensional orthonormal basis, biased towards the labeled data. Moreover, we derived a fixed-point iteration for the pre-image problem for the graph regularized kernel introduced in [Sindhwani et al., 2005] as yet another way of including *a priori* knowledge into the kernel PCA denoising scheme. Experimental results on both simulated data and images from the ALOI database [Geusebroek et al., 2005], demonstrated how the proposed framework leads to improved denoising performance, *i.e.*, the semi-supervised learning technique yields a more descriptive representation of the signal manifold in kernel PCA, and thereby improve the denoising performance compared to classical unsupervised kernel PCA denoising. Future research in this area should focus on quantifying the impact of the model parameters, *i.e.*, the loss term in semi-supervised kernel PCA objective may be interpreted as a locally constraint

similar to the one used in the semi-supervised eigenvector objective [Hansen and Mahoney, 2012], where the related Lagrange multiplier had an interpretation in terms of the teleportation parameter in a diffusion process. Presumably, we can achieve similar interpretations for the semi-supervised kernel PCA parameters, that would yield valuable insights.

Appendix A

Non-parametric Co-clustering of Large Scale Sparse Bipartite Networks on the GPU

Published in *Machine Learning for Signal Processing (MLSP 2011)*

NON-PARAMETRIC CO-CLUSTERING OF LARGE SCALE SPARSE BIPARTITE NETWORKS ON THE GPU

Toke Jansen Hansen, Morten Mørup, Lars Kai Hansen

Section for Cognitive Systems
DTU Informatics
Technical University of Denmark

ABSTRACT

Co-clustering is a problem of both theoretical and practical importance, e.g., market basket analysis and collaborative filtering, and in web scale text processing. We state the co-clustering problem in terms of non-parametric generative models which can address the issue of estimating the number of row and column clusters from a hypothesis space of an infinite number of clusters. To reach large scale applications of co-clustering we exploit that parameter inference for co-clustering is well suited for parallel computing. We develop a generic GPU framework for efficient inference on large scale sparse bipartite networks and achieve a speedup of two orders of magnitude compared to estimation based on conventional CPUs. In terms of scalability we find for networks with more than 100 million links that reliable inference can be achieved in less than an hour on a single GPU. To efficiently manage memory consumption on the GPU we exploit the structure of the posterior likelihood to obtain a decomposition that easily allows model estimation of the co-clustering problem on arbitrary large networks as well as distributed estimation on multiple GPUs. Finally we evaluate the implementation on real-life large scale collaborative filtering data and web scale text corpora, demonstrating that latent mesoscale structures extracted by the co-clustering problem as formulated by the Infinite Relational Model (IRM) are consistent across consecutive runs with different initializations and also relevant for interpretation of the underlaying processes in such large scale networks.

1. INTRODUCTION

Co-clustering also denoted bi-clustering and two-mode clustering is the simultaneous clustering of rows and columns in a matrix. Co-clustering has gained popularity in several fields, including bio-informatics for identification of co-regulated genes and gene functional annotation [20], collaborative filtering and market basket analysis for identification of user and product segments [32], text miming [7, 33] for identification of related terms and documents, and social network modeling to find relationships between agents and behavior [2, 8]. A plethora of co-clustering methods have been proposed over the years including the iterative refinement approach of Hartigan [12], heuristics for grouping columns and rows in an adjacency matrix [31], spectral [7] and information theoretic approaches [6], and methods inspired by community detection in complex networks [33, 30, 8]. For reviews on co-clustering see [21, 20, 8]. Co-clustering can be defined as the matrix decomposition problem

$$\boldsymbol{A} \approx \boldsymbol{R} = \boldsymbol{Z}^{(1)^\top} \boldsymbol{\eta} \boldsymbol{Z}^{(2)},$$

i.e., the data matrix $\boldsymbol{A}$ is decomposed into clusters grouping the row and column entries of the matrix respectively in $\boldsymbol{Z}^{(1)^\top}$ and $\boldsymbol{Z}^{(1)^\top}$ such that the data matrix is segmented into homogenous regions specified by the matrix $\boldsymbol{\eta}$.

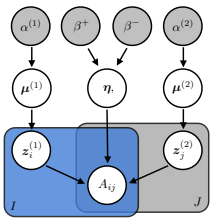
The deviation in reconstruction $D(\boldsymbol{A}||\boldsymbol{R})$ is typically measured in terms of least squares error [12] which corresponds to a Gaussian noise model. For non-negative data the Kullback-Leibler divergence [6] which corresponds to a Poisson noise model has been invoked and for binary data the Bernoulli likelihood [24, 8] has been considered. These noise models result in the following measures of deviation

$$D(\boldsymbol{A}||\boldsymbol{R}) = \begin{cases} \sum_{ij} -\boldsymbol{A}_{ij} \log \frac{\boldsymbol{R}_{ij}}{1-\boldsymbol{R}_{ij}} - \log\left(1-\boldsymbol{R}_{ij}\right) & \text{BERNOULLI} \\ \sum_{ij} \log \boldsymbol{A}_{ij}! - \boldsymbol{A}_{ij} \log \boldsymbol{R}_{ij} + \boldsymbol{R}_{ij} & \text{POISSON} \\ \|\boldsymbol{A}-\boldsymbol{R}\|_F^2 & \text{GAUSSIAN} \end{cases}$$

Two important remaining issues in co-clustering are *scalability*, i.e., to infer latent variables and model parameters for large scale data and *model complexity*, i.e., to determine the number of components of the row and column clusters. For co-clustering this is particularly challenging as the number of components are specified for each mode separately.

In this paper we will exploit that the alternating estimation of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ keeping the remaining parameters fixed parallelizes over the rows and columns respectively of the data matrix to do efficient inference on Graphical Processing Units (GPU). This property has previously been explored in [27, 19] for MapReduce and in cloud environments using FREERIDE [29]. We will approach the co-clustering problem in terms of generative models that serve the two main purposes in machine learning namely to provide interpretation and domain insight via the inference of relevant latent variables as well as identify predictive relations. In particular, we will make use of non-parametric learning that admits to infer the number of row and columns clusters from a hypothesis space of potentially infinitely many clusters through the Chinese Restaurant Process (CRP) [1, 28, 16], this has previously been considered for the Bernoulli likelihood in [33, 16] forming the Infinite Relational Model (IRM).

This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.



	BERNOULLI	POISSON	GAUSSIAN
	$\boldsymbol{\mu}^{(1)} \sim \text{Dirichlet}(\alpha^{(1)}/K^{(1)}\mathbf{1})$	$\boldsymbol{\mu}^{(1)} \sim \text{Dirichlet}(\alpha^{(1)}/K^{(1)}\mathbf{1})$	$\boldsymbol{\mu}^{(1)} \sim \text{Dirichlet}(\alpha^{(1)}/K^{(1)}\mathbf{1})$
	$\boldsymbol{\mu}^{(2)} \sim \text{Dirichlet}(\alpha^{(2)}/K^{(2)}\mathbf{1})$	$\boldsymbol{\mu}^{(2)} \sim \text{Dirichlet}(\alpha^{(2)}/K^{(2)}\mathbf{1})$	$\boldsymbol{\mu}^{(2)} \sim \text{Dirichlet}(\alpha^{(2)}/K^{(2)}\mathbf{1})$
	$\eta_{lm} \sim \text{Beta}(\beta^+, \beta^-)$	$\eta_{lm} \sim \text{Gamma}(\beta^+, \beta^-)$	$\eta_{lm} \sim \text{Normal}(\beta^+, \beta^-)$
	$\boldsymbol{z}_i^{(1)} \sim \text{Discrete}(\boldsymbol{\mu}^{(1)})$	$\boldsymbol{z}_i^{(1)} \sim \text{Discrete}(\boldsymbol{\mu}^{(1)})$	$\boldsymbol{z}_i^{(1)} \sim \text{Discrete}(\boldsymbol{\mu}^{(1)})$
	$\boldsymbol{z}_j^{(2)} \sim \text{Discrete}(\boldsymbol{\mu}^{(2)})$	$\boldsymbol{z}_j^{(2)} \sim \text{Discrete}(\boldsymbol{\mu}^{(2)})$	$\boldsymbol{z}_j^{(2)} \sim \text{Discrete}(\boldsymbol{\mu}^{(2)})$
	$A_{ij} \sim \text{Bernoulli}({\boldsymbol{z}_i^{(1)}}^\top \boldsymbol{\eta} \boldsymbol{z}_j^{(2)})$	$A_{ij} \sim \text{Poisson}({\boldsymbol{z}_i^{(1)}}^\top \boldsymbol{\eta} \boldsymbol{z}_j^{(2)})$	$A_{ij} \sim \text{Normal}({\boldsymbol{z}_i^{(1)}}^\top \boldsymbol{\eta} \boldsymbol{z}_j^{(2)}, \varrho)$

Fig. 1. A graphical representation of the relational model for bipartite graphs. For the GAUSSIAN noise model the additional noise variance parameter ϱ is not illustrated.

1.1. Organization

The paper is organized in the following way: In section 2 we derive the non-parametric co-clustering model based on Bernoulli, Poisson and Gaussian likelihood and show how blocked inference admits efficient parallel computation on the GPU. In section 3 we show a 10^2 speedup of our GPU implementation relative to conventional CPU computing. We analyze four real world large scale networks with up to $\sim 10^9$ non-zero entries in $\boldsymbol{A}$ that pertain to the two important data mining tasks; collaborative filtering and topic modeling, and demonstrate that co-clustering by the IRM on large scale data identify consistent mesoscale structures.

2. METHODS

The graphical model for co-clustering based on the Bernoulli likelihood for binary bipartite graphs, Poisson likelihood for non-negative data and Gaussian noise model is given in figure 1. According to the graphical models each node in mode one and two are assigned to a cluster according to a draw from the discrete distribution parameterized by $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ respectively. $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ are in turn drawn from a Dirichlet distribution parameterized by $\alpha^{(1)}/K^{(1)}\mathbf{1}$ and $\alpha^{(2)}/K^{(2)}\mathbf{1}$. The entry in the matrix $\boldsymbol{A}_{ij}$ is according to the co-clustering model governed by the class assignment of the i^{th} row and j^{th} column, i.e., $\boldsymbol{z}_i^{(1)}$ and $\boldsymbol{z}_j^{(2)}$ as well as the inter-group relations $\boldsymbol{\eta}$ where η_{lm} gives the probability of observing a link between class l and m for the Bernoulli likelihood, the rate or weight for the Poisson likelihood, and the between group mean value of the data matrix for the Gaussian noise model. These inter-group relations η_{lm} are drawn from conjugate priors. Entries in the data matrix $\boldsymbol{A}$ are according to the model conditionally independent given their respective row and column assignments. Finally, assuming an infinite number of clusters is obtained by taking the limits $K^{(1)} \to \infty$ and $K^{(2)} \to \infty$ which has an analytic solution formed by the Chinese Restaurant Process (CRP) [1, 28, 16].

2.1. Blocked Gibbs Sampling

As the Dirichlet distribution is conjugate to the Discrete distribution and the priors on $\boldsymbol{\eta}$ are conjugate both $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}$ and $\boldsymbol{\eta}$ can be integrated out analytically (i.e., collapsed). As a result, inference in the models can be reduced to sequentially sampling each column of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ according to the following

Update $\boldsymbol{Z}^{(1)}$

$$P(z_i^{(1)} = l | \boldsymbol{A}, \alpha^{(1)}, \beta^+, \beta^-) \propto$$

$$\begin{cases} \frac{n_l^{(1)}}{I-1+\alpha^{(1)}} \prod_{l,m} \frac{Beta(N_e(l,m)+\beta^+, N_o(l,m)+\beta^-)}{Beta(\beta^+,\beta^-)} & \text{BERNOULLI} \\ \frac{n_l^{(1)}}{I-1+\alpha^{(1)}} \prod_{l,m} \frac{\Gamma(N_e(\beta^+)(N_t(l,m)+\beta^-)^{N_e(l,m)+\beta^+}}{\Gamma(N_e(l,m)+\beta^+)(\beta^-)^{\beta^+}} & \text{POISSON} \\ \frac{n_m^{(1)}}{I-1+\alpha^{(1)}} \prod_{l,m} \frac{e^{\frac{(N_e(l,m)/\varrho+\beta^+/\beta^-)^2}{2(N_t(l,m)/\varrho+1/\beta^-)} - \frac{\beta^{+2}}{2\beta^-}}}{\sqrt{\beta^-}\sqrt{N_t(l,m)/\varrho+1/\beta^-}} & \text{GAUSSIAN} \end{cases}$$

Update $\boldsymbol{Z}^{(2)}$

$$P(z_j^{(2)} = m | \boldsymbol{A}, \alpha^{(2)}, \beta^+, \beta^-) \propto$$

$$\begin{cases} \frac{n_m^{(2)}}{J-1+\alpha^{(2)}} \prod_{l,m} \frac{Beta(N_e(l,m)+\beta^+, N_o(l,m)+\beta^-)}{Beta(\beta^+,\beta^-)} & \text{BERNOULLI} \\ \frac{n_m^{(2)}}{J-1+\alpha^{(2)}} \prod_{l,m} \frac{\Gamma(N_e(\beta^+)(N_t(l,m)+\beta^-)^{N_e(l,m)+\beta^+}}{\Gamma(N_e(l,m)+\beta^+)(\beta^-)^{\beta^+}} & \text{POISSON} \\ \frac{n_m^{(2)}}{J-1+\alpha^{(2)}} \prod_{l,m} \frac{e^{\frac{(N_e(l,m)/\varrho+\beta^+/\beta^-)^2}{2(N_t(l,m)/\varrho+1/\beta^-)} - \frac{\beta^{+2}}{2\beta^-}}}{\sqrt{\beta^-}\sqrt{N_t(l,m)/\varrho+1/\beta^-}} & \text{GAUSSIAN} \end{cases}$$

where $n_l^{(1)}$ and $n_m^{(2)}$ are the number of nodes already assigned to group l and m while an additional proposal cluster according to the CRP is given by $n_{L+1}^{(1)} = \alpha^{(1)}$ and $n_{M+1}^{(2)} = \alpha^{(2)}$ respectively, whereas

$$N_e(l,m) = \sum_{ij} z_{li}^{(1)} A_{ij} z_{mj}^{(2)}, \quad N_t(l,m) = \sum_{ij} z_{li}^{(1)} z_{mj}^{(2)},$$
$$N_o(l,m) = N_t(l,m) - N_e(l,m).$$

Due to the Dirichlet prior imposed on $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ the model penalize small clusters that are not supported by the data thereby selecting the number of components [22, 16]. The analytic integration of $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}$ and $\boldsymbol{\eta}$ introduces a dependence within each mode such that the assignments of each row in mode one and column in mode two no longer can be updated in parallel. Blocked sampling overcomes this issue while potentially also resulting in better mixing [14, 34]. In blocked Gibbs sampling two approaches are commonly invoked to approximate the CRP; Dirichlet Multinomial Allocation (DMA) [11] and Truncated Stick Breaking construction (TSB) [14]. Whereas DMA sample $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ from

the finite Dirichlet distribution given in figure 1 the TSB sample $\boldsymbol{\mu}$ according to

$$v_l^{(1)} \sim \text{Beta}(1 + n_l^{(1)}, \alpha^{(1)} + \sum_{l+1}^{K^{(1)}} n_l^{(1)}),$$

$$v_m^{(2)} \sim \text{Beta}(1 + n_m^{(1)}, \alpha^{(2)} + \sum_{m+1}^{K^{(2)}} n_m^{(2)}),$$

where $v_{K^{(1)}}^{(1)} = 1$ and $v_{K^{(2)}}^{(2)} = 1$ such that

$$\mu_l^{(1)} = v_l^{(1)} \prod_{l'=1}^{l-1}(1 - v_{l'}^{(1)}), \quad \mu_m^{(2)} = v_m^{(2)} \prod_{m'=1}^{m-1}(1 - v_{m'}^{(2)}).$$

The truncation error becomes insignificant when the model is estimated for large values of $K^{(1)}$ and $K^{(2)}$, see also [14]. We presently consider the blocked sampling based on TSB resulting in the following updates for the model parameters

Update $\boldsymbol{Z}^{(1)}$

$$P(z_i^{(1)} = l | \boldsymbol{A}, \boldsymbol{Z}^{(2)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\eta}) \propto$$

$$\begin{cases} \mu_l^{(1)} e^{\sum_{m,j} \log(\frac{\eta_{lm}}{1-\eta_{lm}}) A_{ij} z_{mj}^{(2)} + \log(1-\eta_{lm}) z_{mj}^{(2)}} & \text{Bernoulli} \\ \mu_l^{(1)} e^{\sum_{m,j} \log(\eta_{lm}) A_{ij} z_{mj}^{(2)} - \eta_{lm} z_{mj}^{(2)}} & \text{Poisson} \\ \mu_l^{(1)} e^{\frac{1}{\varrho^2} \sum_{m,j} 2\eta_{lm} A_{ij} z_{mj}^{(2)} - \eta_{lm}^2 z_{mj}^{(2)}} & \text{Gaussian} \end{cases}$$

Update $\boldsymbol{Z}^{(2)}$

$$P(z_j^{(2)} = m | \boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}) \propto$$

$$\begin{cases} \mu_m^{(2)} e^{\sum_{l,i} \log(\frac{\eta_{lm}}{1-\eta_{lm}}) A_{ij} z_{li}^{(1)} + \log(1-\eta_{lm}) z_{li}^{(1)}} & \text{Bernoulli} \\ \mu_m^{(2)} e^{\sum_{l,i} \log(\eta_{lm}) A_{ij} z_{li}^{(1)} - \eta_{lm} z_{li}^{(1)}} & \text{Poisson} \\ \mu_m^{(2)} e^{\frac{1}{\varrho^2} \sum_{l,i} 2\eta_{lm} A_{ij} z_{li}^{(1)} - \eta_{lm}^2 z_{li}^{(1)}} & \text{Gaussian} \end{cases}$$

Update $\boldsymbol{\eta}$

$$P(\eta_{lm} | \boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \beta^+, \beta^-) \sim$$

$$\begin{cases} \text{Beta}(N_e(l,m) + \beta^+, N_o(l,m) + \beta^-) & \text{Bernoulli} \\ \text{Gamma}(N_e(l,m) + \beta^+, N_t(l,m) + \beta^-) & \text{Poisson} \\ \mathcal{N}(\frac{N_e(l,m)/\varrho + \beta^+/\beta^-}{N_t(l,m)/\varrho + 1/\beta^-}, \frac{1}{N_t(l,m)/\varrho + 1/\beta^-}) & \text{Gaussian} \end{cases}$$

Update $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$

$$\mu_l^{(1)} = v_l^{(1)} \prod_{l'=1}^{l-1}(1 - v_{l'}^{(1)}),$$
$$\text{where} \quad v_l^{(1)} \sim \text{Beta}(1 + n_l^{(1)}, \alpha^{(1)} + \textstyle\sum_{l+1}^{K^{(1)}} n_l^{(1)}),$$

$$\mu_m^{(2)} = v_m^{(2)} \prod_{m'=1}^{m-1}(1 - v_{m'}^{(2)}),$$
$$\text{where} \quad v_m^{(2)} \sim \text{Beta}(1 + n_m^{(2)}, \alpha^{(2)} + \textstyle\sum_{m+1}^{K^{(2)}} n_m^{(2)}).$$

In [34] blocked sampling based on TSB was found to perform as well as collapsed sampling for the Bernoulli likelihood forming the Infinite Relational Model (IRM). We note that while we will use Markov Chain Monte Carlo (MCMC) sampling for parameter inference the approach trivially generalize to variational methods [34]. In particular, a variational implementation follows by taking the expectations over the derived posteriors based on the following factorized distribution of the considered graphical model given in figure 1 of the co-clustering model based on the TSB

$$q(\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \boldsymbol{\eta}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}) = \prod_i q(\boldsymbol{z}_i^{(1)}) \prod_j q(\boldsymbol{z}_j^{(2)}) \prod_{lm} q(\eta_{lm}) \prod_l q(v_l^{(1)}) \prod_m q(v_m^{(2)}).$$

However, we note that the variational inference requires more memory as $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ are no longer hard cluster assignment matrices but dense matrices defining soft clustering.

2.2. The CUDA Architecture

The GPU has evolved into a highly parallel, multithreaded, many-core processor. It is now not specialized for graphics rendering but can also be used for generic intensive parallel computations. Compared to a CPU that is well suited for processing code with a complex control flow, a GPU is much better suited for addressing problems that can be expressed as data-parallel computations with a high arithmetic intensity. CUDA is a set of compiler tools and language extensions developed by NVIDIA that enable programmers to code algorithms for execution on the GPU. At its core, CUDA uses three key abstractions namely thread groups, shared memories, and barrier synchronizations. These are exposed to the programmer as a minimal set of extensions to C/C++. A CUDA capable GPU consists of a set of Multi Processors (MPs) each containing eight Scalar Processors (SPs) and different types of local memories that the SPs may access. All MPs have also access to a large global memory that, compared to their internal memories, is much slower. A problem to be executed on a CUDA capable device, is setup in a grid, where each element in the grid gets assigned to a thread. The grid is then decomposed into blocks that are scheduled onto the MPs with available resources and the assigned MP will in turn schedule the elements of the block onto its eight scalar processors in warps with up to 32 threads. The best performance is obtained when all threads in a warp execute the same instruction and when the total number of threads in the grid is large. This allows for some of the various overhead latencies to be overlapped with arithmetic operations. For a more detailed description about CUDA capable devices we refer to the CUDA programming guide [25] and the reference manual [26].

2.3. GPU Computing Aspects

Our implementation consists of a CPU part for sampling $\boldsymbol{\eta}$, $\boldsymbol{\mu}^{(1)}$, $\boldsymbol{\mu}^{(2)}$ and a GPU part for sampling $\boldsymbol{Z}^{(1)}$, $\boldsymbol{Z}^{(2)}$. The main reason for not keeping everything in GPU code is that only the sampling of the cluster assignment matrices exhibit an arithmetic intensity sufficient for taking advantage of the GPU architecture, see also figure 2 that illustrates how the computations are divided between the CPU and GPU. For simplicity the figure illustrates the case where the entire problem can be stored in GPU memory, but in practice the limited amount of memory forces us to discretize the sampling into suitable sized subproblems where asynchronous operations can be used to hide the associated memory latency when transferring data between the CPU and GPU. One of the greatest challenges in the development of high performance GPU applications is to keep device memory latency low and thereby all threads occupied with arithmetic operations. Current architectures allow

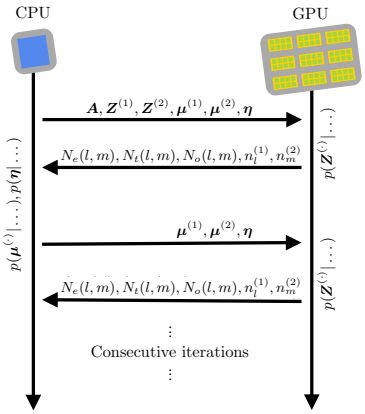


Fig. 2. A simplified view of how model inference is distributed between the CPU and GPU. Initially $\boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}$ are uploaded to the GPU, assuming a sufficient amount of memory. In each iteration, we sample the distributions $p(\boldsymbol{Z}^{(1)}|\dots) \parallel p(\boldsymbol{Z}^{(2)}|\dots)$, compute the sufficient statistics, $N_e(l,m), N_t(l,m), N_o(l,m), n_l^{(1)}, n_m^{(2)}$, and download these to the CPU code. The sufficient statistics are used when sampling $p(\boldsymbol{\mu}^{(1)}|\dots) \parallel p(\boldsymbol{\mu}^{(2)}|\dots) \parallel p(\boldsymbol{\eta}|\dots)$ as seen from the update rules given in section 2.1. In case of sufficient GPU memory, consecutive sampling iterations only upload $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \boldsymbol{\eta}$, otherwise $\boldsymbol{A}, \boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}$ are decomposed into suitable sized blocks that are also uploaded and processed in turn.

memory transfers to be grouped into so-called coalesced memory transfers when all addresses of an executing warp falls within a single memory segment [25]. Unfortunately, sparse matrix representations often result in scattered memory accesses thereby making efficient memory segmentation difficult. Performing sparse matrix operations on CUDA capable GPUs have already been analyzed [3] and with the recent introduction of CUSPARSE the obstacle is indeed simplified, however as the posterior likelihood of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ contain structure that can be exploited to yield a more efficient memory pattern, no existing implementation seems ideal for our purpose. To minimize the memory footprint of the cluster assignment matrices ($\boldsymbol{Z}^{(1)}$, $\boldsymbol{Z}^{(2)}$) we utilize the property that a node can only belong to a single cluster, hence they are encoded as vectors where each entry corresponds to the respective cluster number. With respect to the adjacency matrix $\boldsymbol{A}$, we presort each mode according to node degree and chunk it into aligned blocks of memory where each entry encodes the position of a link. Using the proposed format the calculation of $\boldsymbol{A}\boldsymbol{Z}^{(2)^\top}$ and $\boldsymbol{A}^\top\boldsymbol{Z}^{(1)^\top}$ (constituting the main computational bottleneck in the updates of $\boldsymbol{Z}^{(1)}$ and $\boldsymbol{Z}^{(2)}$ respectively) corresponds to incrementing link counts in the dense result matrix directly indexed by our sparse representation. The implemented memory layout minimizes thread divergence and provides coalesced memory transfers in $\boldsymbol{A}$, whereas the scattered accesses in $\boldsymbol{Z}$ can be cached in local memory. Remaining dense matrix operations are handled using the CUBLAS library, whereas we utilize yet another custom CUDA kernel to sample the cluster assignment matrices done by applying inverse transform sampling requiring a stream of uniform random numbers per thread. Since present devices have no built-in random number generator, pseudo random numbers are generated in GPU code using a hybrid of a Linear Congruential Generator (LCG) and a combined Tausworthe generator that has been shown to reduce the statistical defects observed in each separate generator [23].

3. RESULTS AND DISCUSSION

All experiments have been executed using the following hardware configuration; Intel Core i7-920 2.66GHz with 24 GB of memory, NVIDIA C1060 Tesla with 240 cores and 4 GB memory. Ubuntu 9.10, Linux kernel 2.6.31, NVIDIA CUDA driver version 256.40.

3.1. GPU Performance Investigation

In order to evaluate the speedup of GPU computation over traditional CPU computation we analyze synthetically generated graphs, varying the number of nodes, links and clusters. We highlight that the benchmark compares the GPU code with a semantically equivalent CPU implementation based on the Intel Math Kernel Library (MKL), i.e., the comparison is invariant with respect to the selected likelihood on A_{ij}. Figure 3 shows the speedup which is positively correlated with the number of links and clusters, since increasing these will improve the overall arithmetic intensity on the GPU. Increasing in the number of nodes while keeping the number of links fixed yields a negative impact in the speedup, because increasing the sparsity reduces the load on each parallel processing GPU thread. For 512 clusters our implementation reaches more than a 140 times speedup.

3.2. Large Scale Real Networks

In the analysis of real networks we will focus on co-clustering of binary data, i.e., we consider $A_{ij} \sim \text{Bernoulli}(\boldsymbol{z}_i^{(1)^\top}\boldsymbol{\eta}\boldsymbol{z}_j^{(2)})$ forming the so-called Infinite Relational Model (IRM). This type of relational data is typically represented by the (bipartite) adjacency matrix $\boldsymbol{A} \in \mathbb{B}^{I\times J}$, where $A_{ij} = 1$ if entity i is related to entity j and $A_{ij} = 0$ otherwise. We will consider four datasets, two pertain to collaborative filtering and two to topic modeling.

Collaborative Filtering and Market Basket Data: In market basket analysis consumer purchases of products are stored in a binary adjacency matrix $\boldsymbol{A}$ such that $A_{ij} = 1$ if consumer i purchased product j and $A_{ij} = 0$ otherwise. A benefit of analyzing collaborative filtering/market basket data by co-clustering methods are that consumers and products are automatically divided into segments while unobserved relations can be inferred. We will presently consider consumer purchases of movies based on the Netflix data and user downloads of Facebook applications. The Netflix data[1] contains about 18K movies and 480K consumers purchasing approximately 100 million movies. We note that the collaborative filtering challenge for the Netflix data, i.e., predicting how well users like given movies differ from the current analysis where we predict whether users actually purchased movies. The Facebook dataset of user application consumption was provided

[1] See `http://www.netflixprize.com/`

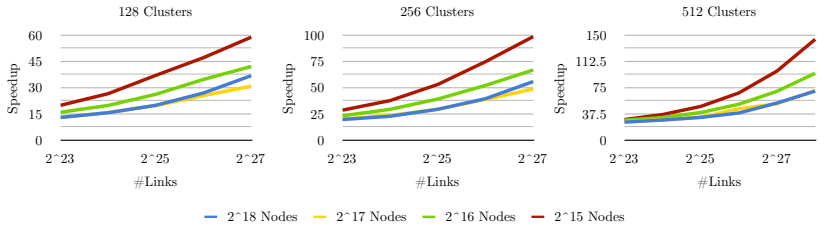


Fig. 3. The speedup in time for GPU computing compared to computing on the CPU when varying the number of links and nodes of the graph. The number of nodes for the two modes are here identical. Left panel; speedup for 128 clusters in each mode, middle panel; speedup for 256 clusters in each mode, right panel; speedup for 512 clusters in each mode.

by [10] and contains about 14K applications and 300K users with a total of almost 6 million installed applications.

Topic Modeling Data: Topic models are statistical models used to discover the abstract topics that occur in a collection of documents. The most well known approaches to topic modeling are the generalization of Latent Semantic Indexing [5] to probabilistic Latent Semantic Indexing (pLSI) [13] and Latent Dirichlet Allocation (LDA) [4]. While these approaches discover topics in documents they also rate within documents how likely specific words are to occur based on some derived measure. We will presently only consider whether a given word occurred in a document or not. As such, co-clustering by the IRM will group documents and terms based on their co-occurrence rather than taking into account the specific number of times given words occurred in the documents, this alleviates any normalization step in the topic modeling. For modeling topics by the IRM we consider articles from New York Times (NYTimes) and abstracts from U.S. National Library of Medicine (PubMed) based on the bag of words representation publicly available from [9]. After tokenization and removal of stop-words the vocabulary of unique words was truncated by only keeping words that occurred more than ten times. Disregarding the number of occurrences of the words we obtain the adjacency matrix $\boldsymbol{A}$ where $\boldsymbol{A}_{ij} = 1$ if word i occurred in document j and $\boldsymbol{A}_{ij} = 0$ otherwise. The New York Times data contains approximately 100K words in 300K documents with approximately 100 million word occurrences whereas the PubMed abstracts contain 140K words and 8 million documents with approximately 500 million word occurrences.

We analyzed all networks based on a maximum of 500 clusters and treated 1 % of links and an equivalent number of non-links as missing at random and analyzed each graph five times with different randomly generated missing sets. This strategy allows us to evaluate the link predictive performance of the IRM. For all analyzed networks convergence was achieved after approximately 100 sampling iterations measured in terms of link predictive performance, therefore all analyses were terminated upon 500 sampling iterations. Table 1 shows the number of extracted components for the two modes as well as the normalized mutual information NMI $= \frac{2 \cdot MI(\hat{\boldsymbol{Z}}, \bar{\boldsymbol{Z}})}{H(\hat{\boldsymbol{Z}}, \hat{\boldsymbol{Z}}) + H(\bar{\boldsymbol{Z}}, \bar{\boldsymbol{Z}})}$ measuring how similar the extracted clusters are across the five sampling runs, i.e., NMI $= 1$ indicates that the cluster structures are identical whereas NMI $= 0$ indicates that there is no information shared between the identified clusters. From the table it can be seen that the extracted clusters all are very robust, i.e., the mutual information is close to saturation given by the mean information in the marginals. In the table we also illustrate link predictability by the area under curve (AUC) of the receiver operator characteristic (ROC) which has been widely used for link prediction in graphs [18, 17]. According to the AUC scores the estimated models are predicting links and non-links significantly better than random guessing (AUC=0.5). In terms of interpretability a clear pattern emerged for the Netflix data where sequels of programs and movies are grouped together. From the term group in NYTimes many highly interpretable topics were extracted, for instance, the IRM extracted topics corresponding to "9/11", "Harry Potter" and "The war on terror", whereas from the PubMed data clusters relating to various areas of research of medicine clearly emerged, while words with the same stem were grouped together.

4. CONCLUSION

We considered the co-clustering problem in terms of non-parametric generative models and demonstrated how a generic GPU implementation admits efficient large scale inference resulting in an order of 10^2 speedup compared to inference on conventional CPUs. For the Bernoulli likelihood the derived co-clustering method is equivalent to the Infinite Relational Model (IRM) of [16, 33, 34] and we found that the model is able to robustly extract interpretable mesoscale information from large scale bipartite networks, providing additional insights into the considered application domains.

A limitation of co-clustering is that it does not directly result in a ranking of the items within a group. We envision several approaches to rank products or terms within a given segment. Future work could be to invoke association mining based on the Apriori algorithm within the extracted segment or to extend the co-clustering model to explicitly rank the entities within a group by introducing a parameter that models the degree in which each entity belong to its assigned cluster. This has been considered for the Poisson likelihood in [15] and readily extends to the present non-parametric co-clustering formulation.

In terms of scalability the current GPU implementation allows us to go beyond the datasets analyzed here and make inference in even larger bipartite networks, provided that $\boldsymbol{Z}^{(1)}$ can fit in device memory when sampling subsets of $\boldsymbol{Z}^{(2)}$ and vice versa. Furthermore, the discretization that we apply in our current implementation makes it straightforward to utilize systems with multi-

Name	I	J	N	$K^{(1)}$	$K^{(2)}$	NMI$^{(1)}$	NMI$^{(2)}$	AUC
Facebook	13604	297474	6 mio.	85.00 (3.74)	117.40 (8.65)	0.74 (3) [0.04(0)]	0.42 (1) [0.01(0)]	0.8117 (66)
Netflix	17770	480189	100 mio	165.20 (16.71)	121.20 (18.02)	0.76 (1) [0.11(1)]	0.69 (1) [0.00(0)]	0.8010 (35)
New York Times	101636	299752	100 mio.	167.40 (20.44)	348.40 (7.20)	0.72 (1) [0.02(0)]	0.73 (1) [0.04(0)]	0.7737 (44)
PubMed	141043	8.2 mio.	483 mio.	488.67 (4.73)	498.67 (0.58)	0.73 (0) [0.07(0)]	0.67 (0) [0.00(0)]	0.7979 (159)

Table 1. The size of the first and second mode of the adjacency matrix $\boldsymbol{A} \in \mathbb{B}^{I \times J}$, the number of non-zero entries $N = |\boldsymbol{A}|$ rounded to closest million as well as the number of extracted clusters for each mode ($K^{(1)}$ and $K^{(2)}$) and the stability of the clusters as measured by normalized mutual information (NMI) across the sampling runs for each mode. In parenthesis is given the standard deviation on last digit across the five runs and in brackets the NMI obtained by random. Link prediction is measured in terms of AUC scores averaged over *low*, *mid*, and *high* node degree regions. We applied this scheme since randomly sampling links and non-links from the graph resulted in a trivially high AUC score. In general we found that high degree nodes were less predictable than low degree nodes across the datasets.

ple GPUs, simply by parallel distribution of the discretized subproblems, i.e., a heterogeneous setup with different levels of parallelism.

5. REFERENCES

[1] C. E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 1974.

[2] M. J. Barber, M. Faria, L. Streit, and O. Strogan. Searching for communities in bipartite networks. AIP, 2008.

[3] N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. Nvidia technical report, 2008.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J. of the American Society for Information Science*, 1990.

[6] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. *KDD*, 2003.

[7] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 2001.

[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 2010.

[9] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[10] M. Gjoka, M. Sirivianos, A. Markopoulou, and X. Yang. Poking facebook: Characterization of osn applicatios. In *WOSN*, 2008.

[11] P. J. Green and S. Richardson. Modelling heterogeneity with and without the dirichlet process. *Board of the Foundation of the Scandinavian Journal of Statistics*, 2001.

[12] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 1972.

[13] T. Hofmann. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, 1999.

[14] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *JASA*, 2001.

[15] B. Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review*, 2011.

[16] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, 2006.

[17] J. Kunegis, E. De Luca, and S. Albayrak. The link prediction problem in bipartite networks. In *Computational Intelligence for Knowledge-Based Systems Design*. 2010.

[18] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 2007.

[19] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. WWW, 2010.

[20] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2004.

[21] I. V. Mechelen, H. H. Bock, and P. D. Boeck. Two-mode clustering methods: a structured overview. *Statistical methods in medical research*, Oct. 2004.

[22] R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *J.of CGS*, 2000.

[23] H. Nguyen. *GPU Gems 3*. 2007.

[24] K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 2001.

[25] NVIDIA. *NVIDIA CUDA Programming Guide 2.0*. 2008.

[26] NVIDIA. *NVIDIA CUDA Reference Manual 2.0*. 2008.

[27] S. Papadimitriou and J. Sun. Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM*, 2008.

[28] J. Pitman. Combinatorial stochastic processes. 2002.

[29] V. Ramanathan, W. Ma, V. T. Ravi, T. Liu, and G. Agrawal. Parallelizing an information theoretic co-clustering algorithm using a cloud middleware. *DMW*, 2010.

[30] J. Reichardt and S. Bornholdt. Clustering of sparse data via network communitiesa prototype study of a large online market. *Journal of Statistical Mechanics*, 2007.

[31] A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. *Handbook of Comp. Molecular Biology*, 2004.

[32] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *In SIGMOD*, 2002.

[33] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Learning infinite hidden relational models. *Uncertainity in Artificial Intelligence (UAI2006)*, 2006.

[34] S. Y. K. Y. H.-P. K. Z. Xu, V.Tresp. Fast inference in inifinite hidden relational models. *Mining and Learning with Graphs (MLG'07)*, 2007.

Appendix B

A Randomized Heuristic for Kernel Parameter Selection with Large-scale Multi-class Data

---

Published in *Machine Learning for Signal Processing (MLSP 2011)*

A RANDOMIZED HEURISTIC FOR KERNEL PARAMETER SELECTION WITH LARGE-SCALE MULTI-CLASS DATA

Toke Jansen Hansen, Trine Julie Abrahamsen, Lars Kai Hansen

Section for Cognitive Systems
DTU Informatics
Technical University of Denmark

ABSTRACT

Over the past few years kernel methods have gained a tremendous amount of attention as existing linear algorithms can easily be extended to account for highly non-linear data in a computationally efficient manner. Unfortunately most kernels require careful tuning of intrinsic parameters to correctly model the distribution of the underlying data. For large-scale problems the multiplicative scaling in time complexity imposed by introducing free parameters in a cross-validation setup will prove computationally infeasible, often leaving pure ad-hoc estimates as the only option. In this contribution we investigate a novel randomized approach for kernel parameter selection in large-scale multi-class data. We fit a minimum enclosing ball to the class means in Reproducing Kernel Hilbert Spaces (RKHS), and use the radius as a quality measure of the space, defined by the kernel parameter. We apply the developed algorithm to a computer vision paradigm where the objective is to recognize 72.000 objects among 1.000 classes. Compared to other distance metrics in the RKHS we find that our randomized approach provides better results together with a highly competitive time complexity.

1. INTRODUCTION

Kernel based classification algorithms account for non-linearities in a computational sophisticated manner through use of the *kernel trick*. Robust selection of intrinsic kernel parameters involves a grid search combined with cross-validation (CV), but for large-scale multi-class data CV becomes both time consuming and resource intensive due to the multiplicative scaling in time complexity imposed by free parameters.

Only few attempts to specifically address the challenge of hyperparameter selection for multi-class problems have been made. While generic algorithms for choosing the hyperparameter in multi-class Support Vector Machines (SVM) was suggested in [1], both [2] and [3] aimed at merely reducing the number of train-validation cycles, e.g., by performing CV on a subsample of the data prior to a restricted line search on the full data set. Several other attempts to more computational attractive approximations to K-fold CV have been made for binary classification [4, 5, 6]. However, in [7] it was shown that all of these approximation schemes were inferior to 5-fold CV.

In this contribution we exploit that previous studies on binary classification have shown, how the intercluster distance in feature space and the optimal hyperparameter defining the RKHS correlates [8, 9]. We extend these attempts to multi-class problems where heuristics for good class separation becomes less immediate. In previous work on intercluster distance based measures for choosing the hyperparameter, it was briefly suggested to maximize the mean of the intercluster distances for multi-class problems [8, 9].

We propose a novel algorithm for hyperparameter selection where a Minimum Enclosing Ball (MEB) is used as a measure of the dispersion of cluster means in the RKHS. Hence, we seek the RKHS that maximizes the MEB. A sublinear algorithm for finding the MEB in a finite dimensional input space was introduced by [10]. In this paper, we device a randomized approximation for MEB estimation in the infinite dimensional RKHS, thereby providing competitive time complexities with respect to existing distance metrics in the RKHS. We demonstrate the developed algorithm by considering image classification on the Amsterdam Library of Object Images (ALOI) [11] and compare the performance with the *median*, *mean*, *maximum* and *minimum* distance measures in the RKHS.

In our experiments we focus on the Gaussian kernel, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$, and use a multi-class SVM in a 5-fold CV setting to establish a ground-truth estimate for comparison with the heuristics. However, the developed heuristics trivially generalize to other kernel functions and kernel machines.

This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

1.1. Organization

This paper is organized in the following way: In Section 2 we introduce the theory behind kernel machines together with the derivation of the considered MEB algorithms. In Section 3 we apply the MEB heuristics to a multi-class object recognition problem and compare our novel approach with related heuristics. Finally, Section 4 briefly concludes the paper.

2. THEORY

Let $\mathcal{H}$ be the RKHS associated with the kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \varphi(\boldsymbol{x})^T \varphi(\boldsymbol{x}')$ (see notation[1]), where $\varphi : \mathcal{X} \mapsto \mathcal{H}$ is a possibly non-linear map from the $D_{\mathcal{X}}$-dimensional input space, $\mathcal{X}$, to the $D_{\mathcal{H}}$-dimensional feature space, $\mathcal{H}$, (possibly infinite dimensional). This is known as the *kernel trick* which states that innerproducts in $\mathcal{H}$ can be computed in terms of kernel evaluations in $\mathcal{X}$. For convenience, all kernel evaluations are collected in the kernel matrix, $\boldsymbol{K} \in \mathbb{R}^{N \times N}$.

2.1. Kernel Machines for Classification

Given a set of training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, $\boldsymbol{x} \in \mathbb{R}^{D_{\mathcal{X}}}$, $y \in \{1, -1\}$, the SVM loss function can be expressed as follows[2]:

$$L_{\text{SVM}} = \min_{\boldsymbol{w} \in \mathcal{H}} \sum_{i=1}^{N} \max(0, 1 - y_i \boldsymbol{w}^\top \varphi(\boldsymbol{x}_i)) + \lambda \|\boldsymbol{w}\|^2, \quad (1)$$

where the first term corresponds to the Hinge-loss function. A computational benefit of the SVM is that classification is based on a subset of training samples defining the margin, these samples are also known as *support vectors*. Note that the slack term controlling the width of the margin is given by $C = \frac{1}{\lambda}$. By applying the representer theorem to rewrite $\boldsymbol{w}$ as $\boldsymbol{w} = \sum_{i=1}^N \alpha_i \varphi(\boldsymbol{x}_i)$, the kernel trick is made applicable. For all kernel type machines the decision function can be expressed as a linear combination of kernel evaluations.

Often direct optimization of the hyperparameters will be infeasible due to non-convexities introduced by the kernel, and a Bayesian treatment such as Automatic Relevance Determination (ARD) will prove computationally heavy even for moderate sized problems when analytic integration over the parameter space is intractable.

For binary classifiers such as the loss functions stated in Eq. (1), two general schemes can be applied to accommodate for multiple classes. One approach is to build one-versus-rest classifiers and to choose the class which classifies the test point with greatest margin/probability. Another strategy is to build a set of one-versus-one classifiers, and select the class based on majority voting [12]. This scheme is applied in our experiments. Even though more classifiers must be trained, the latter approach may prove faster, since the training data set for each classifier is much smaller. The immediate advantage of multiple binary classifiers is that averaging over the classifier decisions will most likely reduce the variance.

[1] Bold uppercase letters denote matrices, bold lowercase letters represent column vectors, and non-bold letters denote scalars.

[2] The bias term, b, has been omitted for notational convenience.

2.2. Clustering Geometry in RKHSs

Given a K class problem with N observations, the j^{th} cluster mean in the RKHS is given by:

$$\boldsymbol{m}_j = \frac{1}{N_{\mathcal{S}_j}} \sum_{i \in \mathcal{S}_j} \varphi(\boldsymbol{x}_i),$$

where $\mathcal{S}_j$ denotes the set of observations belonging to class j. For small values of γ relative to the length scale in input space, any kernelized method approach the equivalent linear method because high order terms in the taylor expansion of the RBF kernel becomes insignificant. Hence, in order to account for non-linearities in the data, γ should be increased. However, in the limit, $\gamma \to \infty$, the following holds:

$$\lim_{\gamma \to \infty} \boldsymbol{K} = \boldsymbol{I} \Rightarrow \lim_{\gamma \to \infty} ||\boldsymbol{m}_i - \boldsymbol{m}_j||^2 = \frac{1}{N_{\mathcal{S}_i}} + \frac{1}{N_{\mathcal{S}_j}}.$$

This result implies that all observations become uncorrelated and the mean of each class will approach $\mathbf{0}$ at a rate inversely proportional to the number of samples within that class. A further result of $\boldsymbol{K}$ approaching $\boldsymbol{I}$ is that the variance of any partitioning of the observations approaches 1, making signal extraction infeasible. Decreasing γ from ∞ will introduce off-diagonal contributions in $\boldsymbol{K}$, leading to an increased distance between the means when the cluster assumption holds (cf. [13]). Since the emerging off-diagonal elements of $\boldsymbol{K}$ depend on the distribution in input space, the exact structure of the intercluster distance as a function of γ will be difficult to quantify without an explicit search, hence finding the optimal γ is nontrivial.

2.3. Minimum Enclosing Ball

Let $\boldsymbol{A}$ denote the matrix of cluster means in feature space:

$$\boldsymbol{A} = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_K],$$

where $\boldsymbol{A} \in \mathbb{R}^{D_{\mathcal{H}} \times K}$. As a measure of the class separation in $\mathcal{H}$ we fit a minimum enclosing ball to the cluster means and choose the optimal γ, and thereby RKHS, as the one leading to the largest MEB (see Figure 1). The MEB problem can be formulated as finding the smallest Euclidean ball

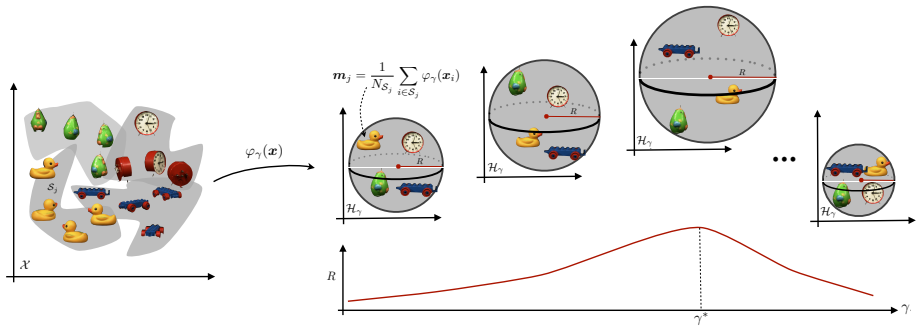


Fig. 1. Illustration of the minimum enclosing ball approach to hyperparameter selection. Each instance is mapped from input space, $\mathcal{X}$, to a RKHS defined by the hyperparameter, γ. The class means are calculated in the RKHS, and the radius, R, of the smallest ball that encloses all cluster means are determined using either the MEB, RMEB or R2MEB algorithm. The optimal hyperparameter is chosen as the one which maximizes the radius of the minimum enclosing ball.

in $D_{\mathcal{H}}$ which contains all columns of $\boldsymbol{A}$, which can be formulated as:

$$\boldsymbol{c}^* = \underset{\varphi(\boldsymbol{x}) \in \mathbb{R}^{D_{\mathcal{H}}}}{\operatorname{argmin}} \max_{i \in [K]} ||\varphi(\boldsymbol{x}) - \boldsymbol{a}_i||^2,$$

where $\max_{i \in [K]} ||\varphi(\boldsymbol{x}) - \boldsymbol{a}_i||^2$ is the radius of the ball, and $\boldsymbol{c}^*$ is the center which minimizes the ball. The above can be reformulated as [10]:

$$\boldsymbol{c}^* = \underset{\varphi(\boldsymbol{x}) \in \mathbb{R}^{D_{\mathcal{H}}}}{\operatorname{argmin}} \max_{\boldsymbol{p} \in \Delta_K} \sum_{i \in [K]} p_i ||\varphi(\boldsymbol{x}) - \boldsymbol{a}_i||^2 \,, \quad (2)$$

where $\Delta_K = \{\boldsymbol{p} \in \mathbb{R}^K | \sum_i p_i = 1,\ p_i \geq 0\}$ is the unit simplex. Thus, maximizing $\boldsymbol{p}$ puts all its weight on the farthest point.

Since we are only interested in finding the radius of the ball, the possible infinite dimensionality of $\boldsymbol{c}^*$ is not of importance. To calculate the distance $\|\varphi(\boldsymbol{x}) - \boldsymbol{a}_i\|^2$ we follow [10] and substitute $\varphi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{p}$ leading to:

$$\|\boldsymbol{A}\boldsymbol{p} - \boldsymbol{a}_i\|^2 = \boldsymbol{p}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{p} + \boldsymbol{a}_i^\top \boldsymbol{a}_i - 2\boldsymbol{p}^\top \boldsymbol{A}^\top \boldsymbol{a}_i \,, \quad (3)$$

where the two terms ($\boldsymbol{a}_i^\top \boldsymbol{a}_i$ and $\boldsymbol{A}^\top \boldsymbol{a}_i$) are simply subsets of the full matrix $\boldsymbol{A}^\top \boldsymbol{A}$. Next we apply the kernel trick:

$$\begin{aligned}(\boldsymbol{A}^\top \boldsymbol{A})_{m,n} &= \frac{1}{N_{\mathcal{S}_m} N_{\mathcal{S}_n}} \sum_{i \in \mathcal{S}_m} \sum_{j \in \mathcal{S}_n} \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j) \\ &= \frac{1}{N_{\mathcal{S}_m} N_{\mathcal{S}_n}} \sum_{i \in \mathcal{S}_m} \sum_{j \in \mathcal{S}_n} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \,. \quad (4)\end{aligned}$$

The dual problem of Eq. (2) can now be used to derive a primal-dual algorithm for optimizing Eq. (2) (cf. [10]).

The algorithm is shown in Alg. 1, where T is the desired number of optimization steps. By running the algorithm for various values of γ, the optimal hyperparameter can be found as $\gamma^* = \underset{\gamma \in \mathbb{R}}{\operatorname{argmax}}\ R(\gamma)$, where R is the radius of the MEB in $\mathcal{H}$.

Algorithm 1 Primal-dual MEB algorithm

1: Let $\boldsymbol{q}^0 \leftarrow [\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}]^\top$ and $\boldsymbol{p}^0 \leftarrow \boldsymbol{q}^0$
2: **for** $t = 1$ to T **do**
3: **for all** $i \in [K]$ **do**
4: $q_i^t = q_i^{t-1} \exp(-||\boldsymbol{A}\boldsymbol{p}^{t-1} - \boldsymbol{a}_i||^2)$ {Eq. (3)-(4)}
5: **end for**
6: $\boldsymbol{q}^t = \boldsymbol{q}^t / ||\boldsymbol{q}^t||$
7: $\boldsymbol{p}^t = (1 - 1/t)\boldsymbol{q}^{t-1} + 1/t \cdot \boldsymbol{q}^t$
8: **end for**
9: $k = \operatorname{argmax}\ \boldsymbol{q}^T$
10: $R = ||\boldsymbol{A}\boldsymbol{p}^T - \boldsymbol{a}_k||^2$

When deriving the algorithm we exploit that the point corresponding to the largest value of $\boldsymbol{q}$ is the one farthest from the center. Hence, the radius of the ball can be found as the distance between the estimated center, $\boldsymbol{c}^* \approx \boldsymbol{A}\boldsymbol{p}^T$, and the farthest cluster mean, $\boldsymbol{a}_k$. In line 7, we assume that $\boldsymbol{A}$ is invertible, which translates to requiring that the K cluster means span a K-dimensional subspace of $\mathcal{H}$. If $\boldsymbol{A}$ is degenerate, line 7 becomes an approximation. However, since the columns of $\boldsymbol{A}$ are constructed as linear combinations of the φ-mapped observations this is a fair assumption, since for any positive definite kernel all $\varphi(\boldsymbol{x}_i)$'s are linearly independent as long as $\boldsymbol{x}_i = \boldsymbol{x}_j$ iff $i = j$.

In order to reduce the time complexity, the primal-dual

algorithm can be randomized [10]. Instead of calculating $\boldsymbol{Ap}$, we sample index $j \in [K]$ with probability p_j and substitute $\boldsymbol{Ap}$ with $\boldsymbol{a}_j$. Since j is chosen randomly, $\boldsymbol{a}_j$ is an unbiased estimator of $\boldsymbol{Ap}$. The randomized MEB (RMEB) algorithm is given in Alg. 2.

Algorithm 2 RMEB algortihm

1: Let $\boldsymbol{q}^0 \leftarrow [\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}]^\top$ and $\boldsymbol{p}^0 \leftarrow \boldsymbol{q}^0$
2: **for** $t = 1$ to T **do**
3: **for all** $i \in [K]$ **do**
4: Choose $j^{t-1} \in [K]$ by $j^{t-1} \leftarrow j$ with prob. p_i^{t-1}
5: $q_i^t = q_i^{t-1} \exp(-||\boldsymbol{a}_{j_{t-1}} - \boldsymbol{a}_i||^2)$
6: **end for**
7: $\boldsymbol{q}^t = \boldsymbol{q}^t / ||\boldsymbol{q}^t||$
8: $\boldsymbol{p}^t = (1 - 1/t)\boldsymbol{q}^{t-1} + 1/t\boldsymbol{q}^t$
9: **end for**
10: $k = \text{argmax } \boldsymbol{q}^T$
11: $R = ||\boldsymbol{Ap}^T - \boldsymbol{a}_k||^2$

The time complexity can be reduced further by randomizing the radius estimate in Line 11 of Alg. 2, leading to the randomized radius MEB (R2MEB) algorithm, where the full $\boldsymbol{Ap}^T$ is approximated by choosing index $l \in [K]$ with probability p_l^T and the radius is then estimated as $R = ||\boldsymbol{A}p_l^T - \boldsymbol{a}_k||^2$. For large-scale problems randomization of the radius estimate will lead to a significant speed up, as only one column of $\boldsymbol{Ap}^T$ is calculated. Since this randomization approximates the center of the MEB by a single data point, some variability of the estimate is inevitable, however, by construction the estimate converges in expectation.

Evidently, the radius of the MEB is upperbounded by half of the maximum of the pairwise distances between the cluster means.

2.4. Other Heuristics for Measuring Class Dispersion

Other natural measures of the class seperation include the *minimum, mean, median* or *maximum* of the pairwise distances between the cluster means which can all be found based on the kernel matrix $\boldsymbol{A}^\top \boldsymbol{A}$.

However, these heuristics all suffer from instability in different scenarios. If two cluster means are located very close in the RKHS, the class dispersion measure based on maximizing the minimum distance will collapse. In this case the MEB approach is still robust as long as only a small fraction of the classes are "very close", i.e., the pseudo-inverse of $\boldsymbol{A}$ is still well-defined. In the other extreme, if one class is very distinct from the rest, maximizing the maximum pairwise distance, are not guaranteed to provide good class separation of the remaining classes. However, in this case the MEB approach will not only separate the "odd" class but also optimize the class separation of the more "similar" classes in order to achieve the largest possible minimum enclosing ball. While both the mean and the median is slightly more robust in such scenarios, it will still fail in extreme cases. Finally, a potential issue regarding the median measure is that it allows for large variability in the distances as long as the median distance do not change, thereby not necessarily identifying the optimal class separation for all classes.

In terms of time complexity the considered methods all require $\mathcal{O}(N^2)$ operations for computing the Gram matrix[3], but only $\mathcal{O}(K^2)$ storage since only intercluster distances between class means in the RKHS are required by the heuristics, i.e., we only store $\boldsymbol{A}^\top \boldsymbol{A}$. Since the *mean*, *maximum* and *minimum* distances can be updated in an online fashion, these can be calculated on the fly using simple bookkeeping while computing $\boldsymbol{A}^\top \boldsymbol{A}$, hence, these only add a constant term to the overall time complexity. On the other hand there exist no $\mathcal{O}(1)$ online update for the *median*, so this quantity requires an additional $\mathcal{O}(K \cdot \log(K))$[4] for sorting the distances. For the MEB algorithm the additional time complexity becomes $\mathcal{O}(T \cdot K^2)$ due to the matrix operations in the inner loop of Alg. 1, whereas the RMEB can be computed in $\mathcal{O}(T \cdot K)$. According to our results sufficient convergence was achieved using $T < \log(K)$ iterations, i.e., the RMEB has an average lower time complexity than the *median*. However, for all aforementioned approaches $\mathcal{O}(N^2)$ remains the dominating factor. The coarse approximation exploited by the R2MEB algorithm can further reduce the overall time complexity considerably. Assuming that the classes are fairly balanced, i.e., each class will contain approximately $\frac{N}{K}$ samples, the time complexity for computing the Gram matrix is reduced to $\mathcal{O}(\frac{\min(T,K)}{K} \cdot N^2)$ since elements can be cached and computed in a lazy fashion. Also for this implementation our results show that reasonable results are obtained for $T < \log(K)$.

3. EXPERIMENTS

For efficient computation of all of the heuristics we precompute the tensor $(\boldsymbol{A}^\top \boldsymbol{A})_\gamma \in \mathbb{R}^{K \times K \times \Gamma}$, where the third dimension corresponds to different choices of the hyperparameter, γ, in the Gaussian kernel, selected from the range $\gamma \in \{2^{-6}, 2^{-5}, \ldots, 2^5\}$. The performance of the heuristics are evaluated using image features distorted by uncorrelated gaussian noise having standard deviation $\sigma \in \{0.0, 0.1, 0.2\}$. We compare the heuristics against the multi-class SVM implementation found in LIBSVM [12], where 5-fold cross-validation is applied to obtain a ground truth estimate of both γ and the slack value, $C \in \{10^{-5}, 10^{-4}, \ldots, 10^{10}\}$.

[3] For simplicity we ignore symmetry of $\boldsymbol{A}^\top \boldsymbol{A}$

[4] This is the average time complexity of quicksort.

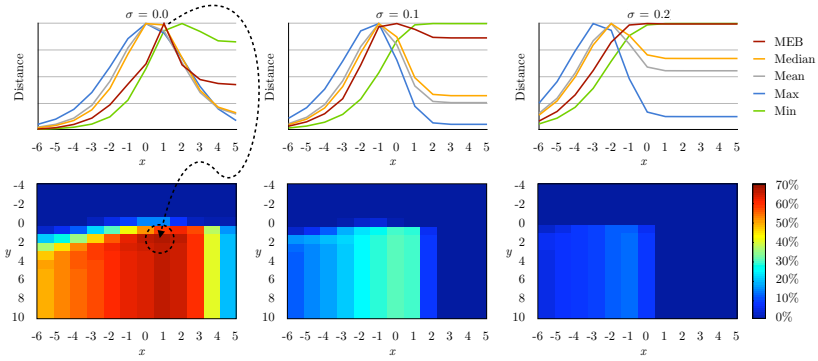


Fig. 2. The panels left to right show the different noise levels ($\sigma = 0.0$, $\sigma = 0.1$ and $\sigma = 0.2$ respectively). The upper panel shows the median, mean, min, and max distances between the class means as well as the radius of the MEB for varying γ values. The results have been normalized for easier comparison. The lower panel shows the 5-fold cross validation accuracy. The parameters are given as $C = 10^y$ and $\gamma = 2^x$. For $\sigma = 0.00$ and $\sigma = 0.1$ it is evident that only the MEB approach peaks at the optimal γ as seen from the CV plot. For $\sigma = 0.2$ the MEB approach saturates at optimal γ value, but no well-defined peak occurs.

3.1. The Amsterdam Library of Object Images

The Amsterdam Library of Object Images (ALOI) is a collection of 1.000 objects that have been recorded for scientific purposes [11]. In the present we consider object classification where the object viewpoint is shifted in steps of 5° yielding a total of 72 images of each object. For each image in the dataset we compute a set of Speeded Up Robust Features (SURF) inspired by the Scale Invariant Feature Transform (SIFT). Both are used to detect and describe local features in images [14, 15]. Since the number of extracted features may vary across the considered images we apply principal component analysis (PCA) to the extracted features of an image, and select the first principal axis to represent the entire image in a compact low dimensional representation.

Even though all of the heuristics can be computed easily for the entire dataset, we restrict our analysis to a subset of 100 objects from the library. This is necessary, since the establishment of a ground-truth estimate by complete SVM CV proves computationally infeasible for more classes.

The results are summarized in Figure 2 and Table 1. Figure 2 shows the results from the heuristics for varying γ as well as the CV results for comparison.The lower panel clearly shows that the performance is very sensitive to the choice of γ. For the three MEB approaches we use $T = 5$ to obtain an algorithm (RMEB) with a strictly lower time complexity than the median. For illustrative purposes we only show the non-randomized MEB approach in Fig. 2. Table 1 show the classification accuracy of the SVM when using the hyperparameter selected by the various heuristics. It is evident that using the MEB approaches for hyperparameter selection leads to better classification for all noise levels.

When comparing the location of the peaks of the class-separation-measures in the upper panel of Figure 2 to the CV results from the SVM in the lower panel, it is evident that using the minimum intercluster distance for hyperparameter selection leads to too large γ-values, while the other standard heuristics all suggest a too small hyperparameter and thereby too linear kernel embeddings. On the contrary, the MEB approach identifies the optimal γ-value for both $\sigma = 0.0$ and $\sigma = 0.1$. In the very noisy setting ($\sigma = 0.2$) the MEB estimate saturates around the optimal γ-value but no significant peak occurs. However, by Occam's razor one could argue to choose the simplest model (smallest γ in this case). Interestingly, the RMEB algorithm is found to actually peak at γ^* for $\sigma = 0.2$, thereby yielding a better classification accuracy than the MEB implementation in this case. This could be caused by different convergence characteristics of the two algorithms.

4. CONCLUSIONS

We have shown how maximizing the radius of the minimum enclosing ball of the cluster means in the RKHS provide a

Noise level	MEB	RMEB	R2MEB	Median	Mean	Max	Min
$\sigma = 0.0$	**67.11 %**	66.04 % (1.18)	65.03 % (2.50)	65.83 %	65.83 %	65.83 %	65.83 %
$\sigma = 0.1$	**31.25 %**	29.58 % (1.88)	28.32 % (2.78)	28.65 %	28.65 %	28.65 %	9.71 %
$\sigma = 0.2$	8.82 %	**12.49 %** (1.33)	11.61 % (1.39)	12.38 %	12.38 %	10.85 %	0.71 %

Table 1. The table shows the classification accuracy of the SVM when using the hyperparameter suggested by the various heuristics on the ALOI dataset for various noise levels. For the three MEB methods five optimization steps were taken ($T = 5$) and for the randomized approaches the standard deviation is given in brackets. For each noise level, the best classification rate is marked in bold. Clearly the MEB approaches lead to more optimal γ-values and thereby higher accuracy of the SVM.

meaningful heuristic for finding the optimal hyperparameter (and hence, RKHS) for kernel machines in multi-class classification problems. Compared to other standard distance metrics in RKHSs we found that the MEB approach provides better results together with a highly competitive time complexity for large scale multi-class data. Under noisy conditions, the performance of the randomized MEB approach indicated a faster convergence of the RMEB than the MEB approach in this setting.

Due to the low time complexity and improved performance, we suggest to use the minimum enclosing ball for crude hyperparameter selection in large-scale problems.

Future work includes testing on a wider range of large scale multi-class classification problems. Furthermore, outlier detection by fitting a MEB in the RKHS is a natural unsupervised extension.

5. REFERENCES

[1] Ana Carolina Lorena and André C. P. L. F. de Carvalho, "Evolutionary tuning of svm parameter values in multiclass problems," *Neurocomput.*, vol. 71, pp. 3326–3334, October 2008.

[2] Matthias Varewyck and Jean-Pierre Martens, "A Practical Approach to Model Selection for Support Vector Machines With a Gaussian Kernel," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 2, pp. 330–340, Apr. 2011.

[3] CJ Van Heerden and E. Barnard, "Towards understanding the influence of svm hyperparameters," in *21st Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, 2010, pp. 69–74.

[4] Thorsten Joachims, *The Maximum-Margin Approach to Learning Text Classifiers:Methods, theory, and algorithms*, Ph.D. thesis, Dortmund University, 2001.

[5] Grace Wahba, *Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV*, pp. 69–88, MIT Press, Cambridge, MA, USA, 1999.

[6] V. Vapnik and O. Chapelle, "Bounds on error expectation for support vector machines," *Neural Comput.*, vol. 12, pp. 2013–2036, September 2000.

[7] Kaibo Duan, S. Sathiya Keerthi, and Aun Neow Poo, "Evaluation of simple performance measures for tuning svm hyperparameters," *Neurocomputing*, vol. 51, pp. 41–59, 2003.

[8] Kuo-Ping Wu and Sheng-De Wang, "Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space," *Pattern Recognition*, vol. 42, pp. 710–717, May 2009.

[9] Song Xiaoshan, Jiang Xiaoyu, Han Chongzhao, and Luo Jianhua, "Inter-class distance based kernel parameter evaluating method for rbf-svm," *Digital Manufacturing and Automation, International Conference on*, vol. 1, pp. 853–858, 2010.

[10] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff, "Sublinear optimization for machine learning," in *FOCS*, 2010, pp. 449–457.

[11] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[12] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[13] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf, "Cluster kernels for semi-supervised learning," in *Neural Information Processing Systems*, 2002, pp. 585–592.

[14] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, pp. 346–359, June 2008.

[15] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, November 2004.

Appendix C

Decoding Complex Cognitive States Online by Manifold Regularization in Real-time fMRI

Published in *Machine Learning and Interpretation in Neuroimaging (MLINI 2011, LNAI 7263, pp. 76-83, 2012)*

Decoding Complex Cognitive States Online by Manifold Regularization in Real-time fMRI

Toke Jansen Hansen[1], Lars Kai Hansen[1] and Kristoffer Hougaard Madsen[2]

[1] Section for Cognitive Systems, DTU Informatics, Technical University of Denmark
[2] Danish Research Centre for Magnetic Resonance Copenhagen University Hospital Hvidovre

Abstract. Human decision making is complex and influenced by many factors on multiple time scales, reflected in the numerous brain networks and connectivity patterns involved as revealed by fMRI.
We address mislabeling issues in paradigms involving complex cognition, by considering a manifold regularizing prior for modeling a sequence of neural events leading to a decision. The method is directly applicable for online learning in the context of real-time fMRI, and our experimental results show that the method can efficiently avoid model degeneracy caused by mislabeling.

1 Introduction

The study of human decision and other higher cognitive functions with fMRI is hampered by several methodological issues including the lack of realism of the experimental situation and lack of interactivity in the decision making process [9]. In particular most experiments involve a predefined set of choices and decision making scenarios. It is well know that open ended active learning protocols can significantly enhance the information extracted in experimental settings and improve the generalizability and learning curve [3]. Working towards active learning protocols in neuroimaging we here discuss the possibility of combining real-time fMRI and online machine learning which will allow experimental interventions dependent on the cognitive state of the subject. Such interventions are crucial for establishing causal relations in the human brain and to study human cognition in general.

1.1 Contributions

Nonlinear algorithms have been used to decode complex cognitive states with an improved generalization performance compared to linear approaches, suggesting that complex interactions in brain patterns are important for distinguishing cognitive states [2, 4–6]. Furthermore, recent experimental results indicate that the manifold assumption is valid for fMRI data, and that augmenting unlabeled resting state data can improve classification performance [1]. Also, spectral methods have successfully been applied in fMRI analyses, e.g., in [8] the intrinsic manifold structure of fMRI data is exploited to construct a low-dimensional graph

of brain states, by embedding the data using a few eigenvectors of graph Laplacian. Moreover, [7] considers a manifold based generative model for inter-subject diffusion maps based on a Gaussian likelihood modeling embedding coordinates, to capture a general atlas of functional connectivity across subjects. Compared to our study they consider coherent and functionally equivalent regions across subject, whereas we consider coherent structures in the intra-subject decision process. In this contribution we build upon nonlinear manifold methods, and address the important challenge of modeling a sequence of neural events leading to a decision. As an example, consider a paradigm where each trial consists of multiple scanned volumes that must be associated with a response variable obtained at the end of each trial. A traditional but likely degenerate approach is to average the samples, or to assume that all samples reflect the same cognitive state, nevertheless that numerous cognitive states can be present during a single trial. To model the intrinsic decision process more naturally we impose a manifold regularizing prior, and thereby rely on the smoothness of fMRI data to infer the brain state of each individual sample. Our primary focus in the following will be on online learning in the context of real-time fMRI, but our findings are also applicable to other settings where the cognitive state can be difficult to explicitly quantify. Figure 3 illustrates the experimental setup of our real-time Brain Computer Interface (BCI) fMRI pipeline.

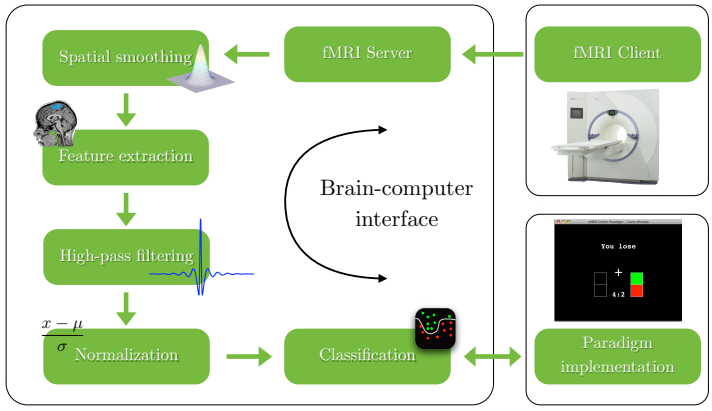


Fig. 1: *Shows the component organization of our Brain Computer Interface (BCI) pipeline. Volumes are streamed over a network connection to a computer responsible for various usual preprocessing stages, where arrows indicate the direction of the flow. The classification stage receives preprocessed fMRI volumes and communicates with the paradigm implementation, by sending predictions and receiving subject events in the form of button presses.*

2 Methods

Given a graph $G = (V, E)$ where the vertices V are the data points, $\boldsymbol{x}_i$, and edges E are represented by an $N \times N$ matrix $\boldsymbol{W}$, i.e., an entry W_{ij} is a weight between node i and j, typically chosen to be the Radial Basis Function (RBF) $W_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$. Hence, $\boldsymbol{W}$ represents the similarity between sampled volumes, i.e., $\boldsymbol{x}_i$ correspond to a sampled fMRI volume represented in a vector space. We form the normalized graph Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, where $\boldsymbol{I}$ is the identity and $\boldsymbol{D}$ is the degree diagonal matrix with elements $D_{ii} = \sum_j W_{ij}$. Using the graph Laplacian we can define a smoothness operator that takes the unlabeled data into account, i.e., we seek functions f that agree with the labeled data but are also smooth with respect to the graph. The smoothness measured by the graph Laplacian is given by $\boldsymbol{f}^\top \boldsymbol{L}\boldsymbol{f}$ that we can view as a zero mean Gaussian process $\boldsymbol{f} \sim \mathcal{N}(0, \boldsymbol{L}^{-1})$.

We model a C class classification problem by considering a likelihood based on a Softmax Function Model (SFM)

$$p(\boldsymbol{z}_i|\boldsymbol{f}_i) = \prod_{j=1}^{C} \left(\frac{\exp(\boldsymbol{f}_i^j)}{\sum_{k=1}^{C} \exp(\boldsymbol{f}_i^k)} \right)^{\boldsymbol{z}_i^j} \tag{1}$$

where $\boldsymbol{f}_i = f(\boldsymbol{x}_i)$ are latent variables generated by the process $\boldsymbol{x} \to \boldsymbol{f} \to \boldsymbol{z}$, and $\boldsymbol{z}_i = [0, \ldots, 0, \boldsymbol{z}_i^j = 1, 0, \ldots, 0]$ encodes that $\boldsymbol{x}_i$ belongs to class j ($j = 1, 2 \ldots, C$), where $\boldsymbol{z}_i = \boldsymbol{0}$ corresponds to an unlabeled sample. Note that the likelihood is only valid for labeled samples, which implies that no probability mass is assigned for remaining unlabeled. To aggregate a trial decision $\boldsymbol{y}_k$, we consider the extended process $\boldsymbol{x} \to \boldsymbol{f} \to \boldsymbol{z} \to \boldsymbol{y}$, and seek a good point estimate of the joint posterior

$$p(\boldsymbol{f}_{i\in\mathcal{K}}, \boldsymbol{\theta}|\boldsymbol{y}_k) \propto \int p(\boldsymbol{y}_k|\boldsymbol{\theta}, \boldsymbol{z}_{i\in\mathcal{K}}) p(\boldsymbol{z}_{i\in\mathcal{K}}|\boldsymbol{f}_{i\in\mathcal{K}}) p(\boldsymbol{f}_{i\in\mathcal{K}}) p(\boldsymbol{\theta}) \partial \boldsymbol{z}_{i\in\mathcal{K}}, \tag{2}$$

where $\mathcal{K}$ is the set of volume indices associated with the k^{th} trial, and $\boldsymbol{\theta}$ is a model parameter. To make the optimization problem tractable, we will approximate the point estimate by first optimize for $\boldsymbol{f}$ in $p(\boldsymbol{z}_{i\in\mathcal{K}}|\boldsymbol{f}_{i\in\mathcal{K}})p(\boldsymbol{f}_{i\in\mathcal{K}})$ based on few relevant labeled samples from each trial, followed by a second classification model, parameterized by $\boldsymbol{\theta}$, used for aggregating $\boldsymbol{z}_{i\in\mathcal{K}}$ to $\boldsymbol{y}_k$, i.e., the trial decision. In the remaining part we focus on optimizing for $\boldsymbol{f}$, and for notational convince we encode

$$\boldsymbol{F}_N = [\boldsymbol{f}_1^1, \boldsymbol{f}_2^1, \ldots, \boldsymbol{f}_N^1, \ldots, \boldsymbol{f}_1^C, \boldsymbol{f}_2^C, \ldots, \boldsymbol{f}_N^C]^\top \tag{3}$$

$$\boldsymbol{Z}_N = [\boldsymbol{z}_1^1, \boldsymbol{z}_2^1, \ldots, \boldsymbol{z}_N^1, \ldots, \boldsymbol{z}_1^C, \boldsymbol{z}_2^C, \ldots, \boldsymbol{z}_N^C]^\top \tag{4}$$

and we then incorporate the prior $\boldsymbol{F}_N \sim \mathcal{N}(\boldsymbol{0}, (\boldsymbol{I}_C \otimes \boldsymbol{L})^{-1})$ into the posterior likelihood, $p(\boldsymbol{F}_N|\boldsymbol{Z}_N) \propto p(\boldsymbol{Z}_N|\boldsymbol{F}_N)p(\boldsymbol{F}_N)$, where the Kronecker product simply corrects for the change in dimensionally caused by the encoding in Eq. 3 and 4.

Because the expression for the posterior likelihood is convex, we optimize $\boldsymbol{F}_N$ by Newton-Raphson to get a MAP estimate that we can then be use to calculate

$\boldsymbol{Z}_N$ by direct substitution into the expression for the SFM, i.e., a transductive step.

$$\Psi(\boldsymbol{F}_N) = -\log p(\boldsymbol{Z}_N|\boldsymbol{F}_N) - \log p(\boldsymbol{F}_N), \quad \boldsymbol{F}_N^{new} = \boldsymbol{F}_N - (\nabla\nabla\Psi)^{-1}\nabla\Psi \tag{5}$$

$$\nabla\Psi = \boldsymbol{\alpha}_N + (\boldsymbol{I}_C \otimes \boldsymbol{L})\boldsymbol{F}_N, \quad \boldsymbol{\alpha}_N = \nabla_{\boldsymbol{F}_N}(-\log p(\boldsymbol{Z}_N|\boldsymbol{F}_N)) \tag{6}$$

$$\nabla\nabla\Psi = \nabla\nabla_{\boldsymbol{F}_N}(-\log p(\boldsymbol{Z}_N|\boldsymbol{F}_N)) + \boldsymbol{I}_C \otimes \boldsymbol{L} \tag{7}$$

To keep things compact we refer to [11] for the derivation of $\nabla_{\boldsymbol{F}_N}(-\log p(\boldsymbol{Z}_N|\boldsymbol{F}_N))$ and $\nabla\nabla_{\boldsymbol{F}_N}(-\log p(\boldsymbol{Z}_N|\boldsymbol{F}_N))$.

We make the model applicable in the online setting by augmenting $\boldsymbol{W}$ when new samples arrive, i.e., updating $\boldsymbol{W}$ is bound by $\mathcal{O}(N)$. Since the square root of a diagonal matrix $\boldsymbol{D}$ is again a diagonal matrix, formed by taking a square root of each of the entries on the diagonal we can write $\boldsymbol{D}^{-1/2} = \mathrm{diag}((\sum_j W_{1j})^{-1/2}, (\sum_j W_{2j})^{-1/2}, \ldots, (\sum_j W_{Nj})^{-1/2})$, hence, updating $\boldsymbol{D}$ is also bound by $\mathcal{O}(N)$. Recalculating the normalized graph Laplacian is bound by $\mathcal{O}(N^2)$ due to $\boldsymbol{D}$ being diagonal, and to reduce the complexity bound $\mathcal{O}(N^3)$ of the consecutive Newton-Raphson iterations, we can maintain a Cholesky factorization of the Hessian in Eq. 7.

2.1 Synthetic Data

To highlight the purpose of our modeling technique we demonstrate the model in greater detail in terms of an easily visualizable data set. We consider an online binary classification task where we assume a fixed stationary probability distribution for flipping labels in a trial. In relation to fMRI data, we assume that samples temporally near the actual decision are trustworthy, i.e., with high probability these reflect the observed decision. Hence, we can think of the samples to reside on two manifolds, one for each decision, and during a trial we receive samples from both manifolds, modeling the decision process.

For comparison we consider the popular support vector machine (SVM) objective, that has proven to yield good generalization performance in a variety of fMRI studies, see for example [6]. Given training data $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, $\boldsymbol{x} \in \mathbb{R}^{D_{\mathcal{X}}}$, $y \in \{1, -1\}$, the SVM objective is given by

$$L_{\mathrm{SVM}} = \min_{\boldsymbol{\theta}\in\mathcal{H}} \sum_{i=1}^{N} \max(0, 1 - y_i\boldsymbol{\theta}^\top\varphi(\boldsymbol{x}_i)) + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{8}$$

and is applicable for the kernel trick $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_j)$, where $\varphi : \mathcal{X} \mapsto \mathcal{H}$ is a possibly nonlinear map from the $D_{\mathcal{X}}$-dimensional input space $\mathcal{X}$ to the $D_{\mathcal{H}}$-dimensional feature space $\mathcal{H}$ [10]. Morover, for both approaches we use an SVM for aggregating the final trial decisions. We consider each trial to be composed of 8 samples, and Figure 2 illustrates three simulations for variations of the mislabeling probabilities and additive Gaussian noise. See the description underneath the figure for more details.

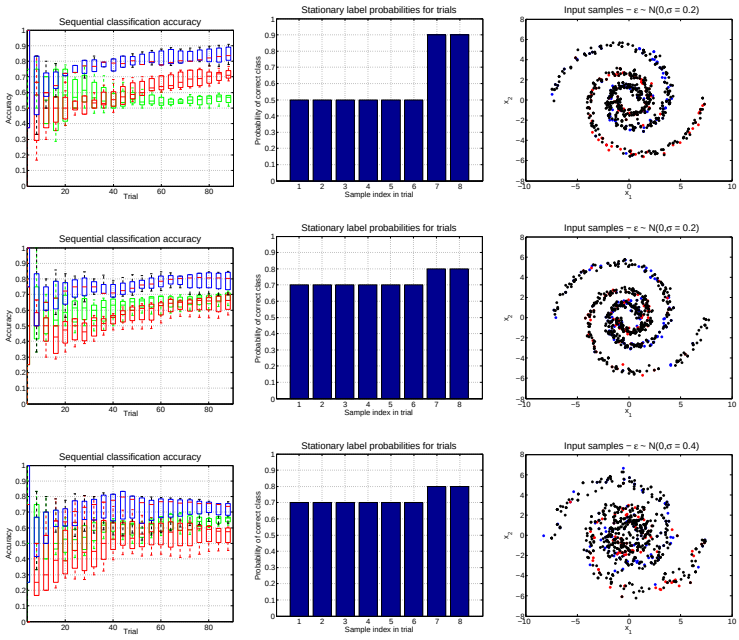


Fig. 2: *Shows three simulations (one per row) for three considered approaches; 1) the semi-supervised approach where we only label the final two samples (blue), 2) an SVM based on all trial samples (green), 3) an SVM based on the final two samples (red). For each simulation the first column shows the accumulated classification accuracy, the second shows the probability for a sample to reflect to the observed trial decision, and the third column shows the observed samples after 100 trials that are used for training of the semi-supervised approach (colors represent labeled samples). Finally, error bars are obtained by resampling, corresponding to a* 95% *confidence interval.*

All approaches will initially produce bad predictions as the intrinsic manifold structure has yet to be learned. However, compared to the other, the semi-supervised model quickly learns the manifold structure and achieves a significantly better classification accuracy for the majority of trials. In the first simulation (top row), it is immediate that the high label uncertainty hurts the SVM approach trained on all trial samples, whereas the SVM trained on the same window as the semi-supervised approach converges much slower, as the unla-

beled samples are not taken into account. In the middle row we reduce the label uncertainty, and as expected the SVM trained on all samples now recovers the structure faster than the one trained on the final two samples. In the bottom row we increase the noise level, and the accuracy gap between the models narrows, as mass can now bleed from one manifold to the other. However, the learning rate of the semi-supervised approach is still better than the others.

3 Brain Imaging Results

We tested the method on Blood Oxygenation Level Dependent (BOLD) sensitive fMRI data acquired on a 3 Tesla MR scanner (Siemens Magnetom Trio). During the scanning session (800 volumes) the subject was engaged in a simple motor paradigm in which the subject was asked to respond by either left or right index finger keypress when a visual cue was presented. The model was used to predict which finger (left or right) the subject selected to press the button with. Preprocessing steps included: rigid body realignment, spatial smoothing (6 mm full width at half maximum isotropic Gaussian kernel), high-pass filtering (cut-off frequency 1/128 Hz), and static masking of premotor cortex. In terms of our approach, we consider a 3 class classification problem, classifying between *baseline*, *left*, and *right*, and as in Section 2.1 we consider the SVM as point of reference. Figure 3 illustrates a conceptual overview of the training approach. We aggregate the trial decision using a kernelized SVM for both approaches. Besides, for both classification stages we applied an RBF kernel, and in the first stage we cross validated in the parameter range given by $\gamma_1 = 2^x, x \in \{-12, -11.9, \ldots, -5\}$, whereas in the next aggregating stage $\gamma_2 = 10^x, x \in \{-10, -9.75, \ldots, 15\}$. We analyzed a single 44 trial scanning session, and measured performance on the final 29 trials, i.e., as we learn and predict in an online fashion we let both approaches stabilize using the first 15 trials. Our method reached a best classification rate of ≈ 0.76, whereas the SVM yields ≈ 0.73, hence we need more data to state significant performance gains, but we do see indications supporting our hypothesis. One reason why the SVM performs relatively well on the ambiguous data may be explained through slackness regularization, but from a modeling perspective the approach is less attractive since mislabeled samples are then treated as outliers, hence, in the SVM mislabeled samples will become support vectors. The suggested "lazy" learning scheme makes few assumptions about the temporal dynamics of the brain state by only assigning hard labels to only a few volumes within each block, while still benefitting from unlabeled samples by identifying the manifold on which the data reside. In essence the suggested classification scheme allows the non-stationary temporal dynamics of a decision process to be captured thereby enabling the identification sequences of related neural events which leads to a decision.

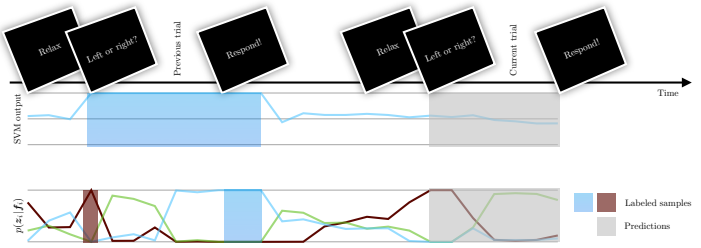


Fig. 3: *Illustrates a conceptual overview (best viewed in color) of the training process of our approach and the SVM used for comparison. The SVM (upper plot) is trained on both the preparation and response samples (highlighted in blue), whereas remaining samples correspond to predictions, i.e., the SVM is trained using samples in this interval through all preceding trials. The semi-supervised approach (lower plot) is trained on a few baseline samples (highlighted in red) and 3 samples around the end of each trial (highlighted in blue), i.e., the three time series correspond to the probability of baseline (red), left (blue) and right (green). Common for both approaches; highlighted in grey are the preparation predictions that must be aggregated into the decision of the current trial, before the subject reveals the actual decision. The classification accuracy is measured as the number of correctly aggregated trial decisions.*

4 Conclusion

In the current work we demonstrated how semi-supervised learning can be used to relax the labeling scheme typically used in brain state classification models based on fMRI data. The suggested lazy labeling scheme makes few assumptions about the temporal dynamics of the brain state by only assigning hard labels to only a few volumes within each block, while still benefitting from unlabeled samples by identifying the manifold on which the data reside. In essence the suggested classification scheme allows the non-stationary temporal dynamics of a decision process to be captured thereby enabling the identification of sequences of related neural events which leads to a decision. Our current results in this preliminary study indicate that the labeling scheme performs on par with existing state of the art nonlinear methods. Future work will focus on comprehensive evaluation and handling of label uncertainty. Furthermore the framework would be particularly for the investigations into how generic the human decision making process is. This could be achieved by investigating the stationarity of $p(\boldsymbol{y}_k|\boldsymbol{\theta}, \boldsymbol{z}_{i\in\mathcal{K}})$ over subjects.

References

1. Blaschko, M., Shelton, J., Bartels, A.: Augmenting feature-driven fmri analyses: Semi-supervised learning and resting state activity. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22, pp. 126–134 (2009)
2. Davatzikos, C., Ruparel, K., Fan, Y., Shen, D.G., Acharyya, M., Loughead, J.W., Gur, R.C., Langleben, D.D.: Classifying spatial patterns of brain activity with machine learning methods: application to lie detection. NeuroImage 28(3), 663–8 (2005), http://www.ncbi.nlm.nih.gov/pubmed/16169252
3. Guyon, I., Cawley, G., Dror, G., Lemaire, V.: Results of the active learning challenge. Journal of Machine Learning Research 16, 19–45 (2011)
4. Haynes, J.D., Sakai, K., Rees, G., Gilbert, S., Frith, C., Passingham, R.E.: Reading hidden intentions in the human brain. Current biology : CB 17(4), 323–8 (2007), http://www.ncbi.nlm.nih.gov/pubmed/17291759
5. Kriegeskorte, N., Goebel, R., Bandettini, P.: Information-based functional brain mapping. Proceedings of the National Academy of Sciences of the United States of America 103(10), 3863–8 (March 2006), http://www.ncbi.nlm.nih.gov/pubmed/16537458
6. LaConte, S.M., Peltier, S.J., Hu, X.P.: Real-time fMRI using brain-state classification. Hum Brain Mapp 28, 1033–1044 (October 2007), http://dx.doi.org/10.1002/hbm.20326
7. Langs, G., Lashkari, D., Sweet, A., Tie, Y., Rigolo, L., Golby, A.J., Golland, P.: Learning an atlas of a cognitive process in its functional geometry. In: Proceedings of the 22nd international conference on Information processing in medical imaging. pp. 135–146. IPMI'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2029686.2029700
8. Meyer, F.: Learning and predicting brain dynamics from fMRI: a spectral approach. vol. 6701, pp. 67010W+. SPIE (2007), http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=PSISDG006701000000167010W000001\&idtype=cvips\&gifs=yes
9. Murawski, C.: Neuroeconomics: Investigating the neurobiology of choice. Australian Economic Review 44(2), 215–224 (2011), http://dx.doi.org/10.1111/j.1467-8462.2011.00638.x
10. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
11. Song, Y., Zhang, C., Lee, J.: Graph based multi-class semi-supervised learning using gaussian process (2006)

Appendix D

Semi-supervised Eigenvectors for Locally-biased Learning

Published in *Neural Information Processing Systems (NIPS 2012)*

Semi-supervised Eigenvectors for Locally-biased Learning

Toke Jansen Hansen
Section for Cognitive Systems
DTU Informatics
Technical University of Denmark
tjha@imm.dtu.dk

Michael W. Mahoney
Department of Mathematics
Stanford University
Stanford, CA 94305
mmahoney@cs.stanford.edu

Abstract

In many applications, one has side information, *e.g.*, labels that are provided in a semi-supervised manner, about a specific target region of a large data set, and one wants to perform machine learning and data analysis tasks "nearby" that pre-specified target region. Locally-biased problems of this sort are particularly challenging for popular eigenvector-based machine learning and data analysis tools. At root, the reason is that eigenvectors are inherently global quantities. In this paper, we address this issue by providing a methodology to construct *semi-supervised eigenvectors* of a graph Laplacian, and we illustrate how these locally-biased eigenvectors can be used to perform *locally-biased machine learning*. These semi-supervised eigenvectors capture successively-orthogonalized directions of maximum variance, conditioned on being well-correlated with an input seed set of nodes that is assumed to be provided in a semi-supervised manner. We also provide several empirical examples demonstrating how these semi-supervised eigenvectors can be used to perform locally-biased learning.

1 Introduction

We consider the problem of finding a set of locally-biased vectors that inherit many of the "nice" properties that the leading nontrivial global eigenvectors of a graph Laplacian have—for example, that capture "slowly varying" modes in the data, that are fairly-efficiently computable, that can be used for common machine learning and data analysis tasks such as kernel-based and semi-supervised learning, etc.—so that we can perform what we will call *locally-biased machine learning* in a principled manner.

By *locally-biased machine learning*, we mean that we have a very large data set, *e.g.*, represented as a graph, and that we have information, *e.g.*, given in a semi-supervised manner, that certain "regions" of the data graph are of particular interest. In this case, we may want to focus predominantly on those regions and perform data analysis and machine learning, *e.g.*, classification, clustering, ranking, etc., that is "biased toward" those pre-specified regions. Examples of this include the following.

- *Locally-biased community identification.* In social and information network analysis, one might have a small "seed set" of nodes that belong to a cluster or community of interest [2, 13]; in this case, one might want to perform link or edge prediction, or one might want to "refine" the seed set in order to find other nearby members.
- *Locally-biased image segmentation.* In computer vision, one might have a large corpus of images along with a "ground truth" set of pixels as provided by a face detection algorithm [7, 14, 15]; in this case, one might want to segment entire heads from the background for all the images in the corpus in an automated manner.

- *Locally-biased neural connectivity analysis.* In functional magnetic resonance imaging applications, one might have small sets of neurons that "fire" in response to some external experimental stimulus [16]; in this case, one might want to analyze the subsequent temporal dynamics of stimulation of neurons that are "nearby," either in terms of connectivity topology or functional response.

These examples present considerable challenges for spectral techniques and traditional eigenvector-based methods. At root, the reason is that eigenvectors are inherently global quantities, thus limiting their applicability in situations where one is interested in very local properties of the data.

In this paper, we provide a methodology to construct what we will call *semi-supervised eigenvectors* of a graph Laplacian; and we illustrate how these locally-biased eigenvectors inherit many of the properties that make the leading nontrivial global eigenvectors of the graph Laplacian so useful in applications. To achieve this, we will formulate an optimization ansatz that is a variant of the usual global spectral graph partitioning optimization problem that includes a natural locality constraint as well as an orthogonality constraint, and we will iteratively solve this problem.

In more detail, assume that we are given as input a (possibly weighted) data graph $G = (V, E)$, an indicator vector s of a small "seed set" of nodes, a *correlation parameter* $\kappa \in [0, 1]$, and a positive integer k. Then, informally, we would like to construct k vectors that satisfy the following bicriteria: first, each of these k vectors is well-correlated with the input seed set; and second, those k vectors describe successively-orthogonalized directions of maximum variance, in a manner analogous to the leading k nontrivial global eigenvectors of the graph Laplacian. (We emphasize that the seed set s of nodes, the integer k, and the correlation parameter κ are part of the input; and thus they should be thought of as being available in a semi-supervised manner.) Somewhat more formally, our main algorithm, Algorithm 1 in Section 3, returns as output k semi-supervised eigenvectors; each of these is the solution to an optimization problem of the form of GENERALIZED LOCALSPECTRAL in Figure 1, and thus each "captures" (say) κ/k of the correlation with the seed set. Our main theoretical result states that these vectors define successively-orthogonalized directions of maximum variance, conditioned on being κ/k-well-correlated with an input seed set s; and that each of these k semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations.

From a technical perspective, the work most closely related to ours is that of Mahoney *et al.* [14]. The original algorithm of Mahoney *et al.* [14] introduced a methodology to construct a locally-biased version of the leading nontrivial eigenvector of a graph Laplacian and showed (theoretically and empirically in a social network analysis application) that the resulting vector could be used to partition a graph in a locally-biased manner. From this perspective, our extension incorporates a natural orthogonality constraint that successive vectors need to be orthogonal to previous vectors. Subsequent to the work of [14], [15] applied the algorithm of [14] to the problem of finding locally-biased cuts in a computer vision application. Similar ideas have also been applied somewhat differently. For example, [2] use locally-biased random walks, *e.g.*, short random walks starting from a small seed set of nodes, to find clusters and communities in graphs arising in Internet advertising applications; [13] used locally-biased random walks to characterize the local and global clustering structure of a wide range of social and information networks; [11] developed the Spectral Graph Transducer (SGT), that performs transductive learning via spectral graph partitioning. The objectives in both [11] and [14] are considered constrained eigenvalue problems, that can be solved by finding the smallest eigenvalue of an asymmetric generalized eigenvalue problem, but in practice this procedure can be highly unstable [8]. The SGT reduces the instabilities by performing all calculations in a subspace spanned by the d smallest eigenvectors of the graph Laplacian, whereas [14] perform a binary search, exploiting the monotonic relationship between a control parameter and the corresponding Lagrange multiplier.

In parallel, [3] and a large body of subsequent work including [6] used eigenvectors of the graph Laplacian to perform dimensionality reduction and data representation, in unsupervised and semi-supervised settings. Many of these methods have a natural interpretation in terms of kernel-based learning [18]. Many of these diffusion-based spectral methods also have a natural interpretation in terms of spectral ranking [21]. "Topic sensitive" and "personalized" versions of these spectral ranking methods have also been studied [9, 10]; and these were the motivation for diffusion-based methods to find locally-biased clusters in large graphs [19, 1, 14]. Our optimization ansatz is a generalization of the linear equation formulation of the PageRank procedure [17, 14, 21], and the solution involves Laplacian-based linear equation solving, which has been suggested as a primitive

of more general interest in large-scale data analysis [20]. Finally, the form of our optimization problem has similarities to other work in computer vision applications: *e.g.*, [23] and [7] find good conductance clusters subject to a set of linear constraints.

2 Background and Notation

Let $G = (V, E, w)$ be a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges, in which edge $\{i, j\}$ has non-negative weight w_{ij}. In the following, $A_G \in \mathbb{R}^{V \times V}$ will denote the adjacency matrix of G, while $D_G \in \mathbb{R}^{V \times V}$ will denote the diagonal degree matrix of G, *i.e.*, $D_G(i,i) = d_i = \sum_{\{i,j\} \in E} w_{ij}$, the weighted degree of vertex i. Moreover, for a set of vertices $S \subseteq V$ in a graph, the *volume of* S is $\mathrm{vol}(S) \stackrel{\text{def}}{=} \sum_{i \in S} d_i$. The Laplacian of G is defined as $L_G \stackrel{\text{def}}{=} D_G - A_G$. (This is also called the combinatorial Laplacian, in which case the normalized Laplacian of G is $\mathcal{L}_G \stackrel{\text{def}}{=} D_G^{-1/2} L_G D_G^{-1/2}$.)

The Laplacian is the symmetric matrix having quadratic form $x^T L_G x = \sum_{ij \in E} w_{ij}(x_i - x_j)^2$, for $x \in \mathbb{R}^V$. This implies that L_G is positive semidefinite and that the all-one vector $1 \in \mathbb{R}^V$ is the eigenvector corresponding to the smallest eigenvalue 0. The generalized eigenvalues of $L_G x = \lambda_i D_G x$ are $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. We will use v_2 to denote smallest non-trivial eigenvector, *i.e.*, the eigenvector corresponding to λ_2; v_3 to denote the next eigenvector; and so on. Finally, for a matrix A, let A^+ denote its (uniquely defined) Moore-Penrose pseudoinverse. For two vectors $x, y \in \mathbb{R}^n$, and the degree matrix D_G for a graph G, we define the degree-weighted inner product as $x^T D_G y \stackrel{\text{def}}{=} \sum_{i=1}^n x_i y_i d_i$. In particular, if a vector x has unit norm, then $x^T D_G x = 1$. Given a subset of vertices $S \subseteq V$, we denote by 1_S the indicator vector of S in $\mathbb{R}^V$ and by 1 the vector in $\mathbb{R}^V$ having all entries set equal to 1.

3 Optimization Approach to Semi-supervised Eigenvectors

3.1 Motivation for the Program

Recall the optimization perspective on how one computes the leading nontrivial global eigenvectors of the normalized Laplacian $\mathcal{L}_G$. The first nontrivial eigenvector v_2 is the solution to the problem GLOBALSPECTRAL that is presented on the left of Figure 1. Equivalently, although GLOBALSPECTRAL is a non-convex optimization problem, strong duality holds for it and it's solution may be computed as v_2, the leading nontrivial generalized eigenvector of L_G. The next eigenvector v_3 is the solution to GLOBALSPECTRAL, augmented with the constraint that $x^T D_G v_2 = 0$; and in general the t^{th} generalized eigenvector of L_G is the solution to GLOBALSPECTRAL, augmented with the constraints that $x^T D_G v_i = 0$, for $i \in \{2, \ldots, t-1\}$. Clearly, this set of constraints and the constraint $x^T D_G 1 = 0$ can be written as $x^T D_G Q = 0$, where 0 is a $(t-1)$-dimensional all-zeros vector, and where Q is an $n \times (t-1)$ orthogonal matrix whose i^{th} column equals v_i (where $v_1 = 1$, the all-ones vector, is the first column of Q).

Also presented in Figure 1 is LOCALSPECTRAL, which includes a constraint requiring the solution to be well-correlated with an input seed set. This LOCALSPECTRAL optimization problem was introduced in [14], where it was shown that the solution to LOCALSPECTRAL may be interpreted as a locally-biased version of the second eigenvector of the Laplacian. In particular, although LOCALSPECTRAL is not convex, it's solution can be computed efficiently as the solution to a set of linear equations that generalize the popular Personalized PageRank procedure; in addition, by performing a sweep cut and appealing to a variant of Cheeger's inequality, this locally-biased eigenvector can be used to perform locally-biased spectral graph partitioning [14].

3.2 Our Main Algorithm

We will formulate the problem of computing semi-supervised vectors in terms of a primitive optimization problem of independent interest. Consider the GENERALIZED LOCALSPECTRAL optimization problem, as shown in Figure 1. For this problem, we are given a graph $G = (V, E)$, with associated Laplacian matrix L_G and diagonal degree matrix D_G; an indicator vector s of a small

GLOBALSPECTRAL	LOCALSPECTRAL	GENERALIZED LOCALSPECTRAL
minimize $x^T L_G x$	minimize $x^T L_G x$	minimize $x^T L_G x$
s.t $x^T D_G x = 1$	s.t $x^T D_G x = 1$	s.t $x^T D_G x = 1$
$x^T D_G 1 = 0$	$x^T D_G 1 = 0$	$x^T D_G Q = 0$
	$x^T D_G s \geq \sqrt{\kappa}$	$x^T D_G s \geq \sqrt{\kappa}$

Figure 1: Left: The usual GLOBALSPECTRAL partitioning optimization problem; the vector achieving the optimal solution is v_2, the leading nontrivial generalized eigenvector of L_G with respect to D_G. Middle: The LOCALSPECTRAL optimization problem, which was originally introduced in [14]; for $\kappa = 0$, this coincides with the usual global spectral objective, while for $\kappa > 0$, this produces solutions that are biased toward the seed vector s. Right: The GENERALIZED LOCALSPECTRAL optimization problem we introduce that includes both the locality constraint and a more general orthogonality constraint. Our main algorithm for computing semi-supervised eigenvectors will iteratively compute the solution to GENERALIZED LOCALSPECTRAL for a sequence of Q matrices. In all three cases, the optimization variable is $x \in \mathbb{R}^n$.

"seed set" of nodes; a *correlation parameter* $\kappa \in [0, 1]$; and an $n \times \nu$ constraint matrix Q that may be assumed to be an orthogonal matrix. We will assume (without loss of generality) that s is properly normalized and orthogonalized so that $s^T D_G s = 1$ and $s^T D_G 1 = 0$. While s can be a general unit vector orthogonal to 1, it may be helpful to think of s as the indicator vector of one or more vertices in V, corresponding to the target region of the graph.

In words, the problem GENERALIZED LOCALSPECTRAL asks us to find a vector $x \in \mathbb{R}^n$ that minimizes the variance $x^T L_G x$ subject to several constraints: that x is unit length; that x is orthogonal to the span of Q; and that x is $\sqrt{\kappa}$-well-correlated with the input seed set vector s. In our application of GENERALIZED LOCALSPECTRAL to the computation of semi-supervised eigenvectors, we will iteratively compute the solution to GENERALIZED LOCALSPECTRAL, updating Q to contain the already-computed semi-supervised eigenvectors. That is, to compute the first semi-supervised eigenvector, we let $Q = 1$, *i.e.*, the n-dimensional all-ones vector, which is the trivial eigenvector of L_G, in which case Q is an $n \times 1$ matrix; and to compute each subsequent semi-supervised eigenvector, we let the columns of Q consist of 1 and the other semi-supervised eigenvectors found in each of the previous iterations.

To show that GENERALIZED LOCALSPECTRAL is efficiently-solvable, note that it is a quadratic program with only one quadratic constraint and one linear equality constraint. In order to remove the equality constraint, which will simplify the problem, let's change variables by defining the $n \times (n-\nu)$ matrix F as $\{x : Q^T D_G x = 0\} = \{x : x = Fy\}$. That is, F is a span for the null space of Q^T; and we will take F to be an orthogonal matrix. Then, with respect to the y variable, GENERALIZED LOCALSPECTRAL becomes

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & y^T F^T L_G F y \\ \text{subject to} \quad & y^T F^T D_G F y = 1, \\ & y^T F^T D_G s \geq \sqrt{\kappa}. \end{aligned} \tag{1}$$

In terms of the variable x, the solution to this optimization problem is of the form

$$\begin{aligned} x^* &= cF \left(F^T (L_G - \gamma D_G) F\right)^+ F^T D_G s \\ &= c \left(F F^T (L_G - \gamma D_G) F F^T\right)^+ D_G s, \end{aligned} \tag{2}$$

for a normalization constant $c \in (0, \infty)$ and for some γ that depends on $\sqrt{\kappa}$. The second line follows from the first since F is an $n \times (n-\nu)$ orthogonal matrix. This so-called "S-procedure" is described in greater detail in Chapter 5 and Appendix B of [4]. The significance of this is that, although it is a non-convex optimization problem, the GENERALIZED LOCALSPECTRAL problem can be solved by solving a linear equation, in the form given in Eqn. (2).

Returning to our problem of computing semi-supervised eigenvectors, recall that, in addition to the input for the GENERALIZED LOCALSPECTRAL problem, we need to specify a positive integer k that indicates the number of vectors to be computed. In the simplest case, we would assume that

we would like the correlation to be "evenly distributed" across all k vectors, in which case we will require that each vector is $\sqrt{\kappa/k}$-well-correlated with the input seed set vector s; but this assumption can easily be relaxed, and thus Algorithm 1 is formulated more generally as taking a k-dimensional vector $\kappa = [\kappa_1, \ldots, \kappa_k]^T$ of correlation coefficients as input.

To compute the first semi-supervised eigenvector, we will let $Q = 1$, the all-ones vector, in which case the first nontrivial semi-supervised eigenvector is

$$x_1^* = c\,(L_G - \gamma_1 D_G)^+ D_G s, \tag{3}$$

where γ_1 is chosen to saturate the part of the correlation constraint along the first direction. (Note that the projections FF^T from Eqn. (2) are not present in Eqn. (3) since by design $s^T D_G 1 = 0$.) That is, to find the correct setting of γ_1, it suffices to perform a binary search over the possible values of γ_1 in the interval $(-\mathrm{vol}(G), \lambda_2(G))$ until the correlation constraint is satisfied, that is, until $(s^T D_G x)^2$ is sufficiently close to κ_1^2, see [8, 14].

To compute subsequent semi-supervised eigenvectors, *i.e.*, at steps $t = 2, \ldots, k$ if one ultimately wants a total of k semi-supervised eigenvectors, then one lets Q be the $n \times (t-1)$ matrix with first column equal to 1 and with j^{th} column, for $i = 2, \ldots, t-1$, equal to x_{j-1}^* (where we emphasize that x_{j-1}^* is a vector not an element of a vector). That is, Q is of the form $Q = [1, x_1^*, \ldots, x_{t-1}^*]$, where x_i^* are successive semi-supervised eigenvectors, and the projection matrix FF^T is of the form $FF^T = I - D_G Q(Q^T D_G D_G Q)^{-1} Q^T D_G$, due to the degree-weighted inner norm. Then, by Eqn. (2), the t^{th} semi-supervised eigenvector takes the form

$$x_t^* \quad = \quad c\left(FF^T(L_G - \gamma_t D_G)FF^T\right)^+ D_G s. \tag{4}$$

Algorithm 1 Semi-supervised eigenvectors

Input: $L_G, D_G, s, \kappa = [\kappa_1, \ldots, \kappa_k]^T, \epsilon$
Require: $s^T D_G 1 = 0$, $s^T D_G s = 1$, $\kappa^T 1 \leq 1$
$Q = [1]$
for $t = 1$ to k **do**
  $FF^T \leftarrow I - D_G Q(Q^T D_G D_G Q)^{-1} Q^T D_G$
  $\top \leftarrow \lambda_2$ where $FF^T L_G FF^T v_2 = \lambda_2 FF^T D_G FF^T v_2$
  $\perp \leftarrow -\mathrm{vol}(G)$
  repeat
    $\gamma_t \leftarrow (\perp + \top)/2$ (Binary search over γ_t)
    $x_t \leftarrow (FF^T(L_G - \gamma_t D_G)FF^T)^+ FF^T D_G s$
    Normalize x_t such that $x_t^T D_G x_t = 1$
    if $(x_t^T D_G s)^2 > \kappa_t$ **then** $\perp \leftarrow \gamma_t$ **else** $\top \leftarrow \gamma_t$ **end if**
  until $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$ **or** $\|(\perp + \top)/2 - \gamma_t\| \leq \epsilon$
  Augment Q with x_t^* by letting $Q = [Q, x_t^*]$.
end for

In more detail, Algorithm 1 presents pseudo-code for our main algorithm for computing semi-supervised eigenvectors. Several things should be noted about our implementation. First, note that we implicitly compute the projection matrix FF^T. Second, a naïve approach to Eqn. (2) does not immediately lead to an efficient solution, since $D_G s$ will not be in the span of $(FF^T(L_G - \gamma D_G)FF^T)$, thus leading to a large residual. By changing variables so that $x = FF^T y$, the solution becomes $x^* \propto FF^T(FF^T(L_G - \gamma D_G)FF^T)^+ FF^T D_G s$. Since FF^T is a projection matrix, this expression is equivalent to $x^* \propto (FF^T(L_G - \gamma D_G)FF^T)^+ FF^T D_G s$. Third, we exploit that $FF^T(L_G - \gamma_i D_G)FF^T$ is an SPSD matrix, and we apply the conjugate gradient method, rather than computing the explicit pseudoinverse. That is, in the implementation we never represent the dense matrix FF^T, but instead we treat it as an operator and we simply evaluate the result of applying a vector to it on either side. Fourth, we use that λ_2 can never decrease (here we refer to λ_2 as the smallest non-zero eigenvalue of the modified matrix), so we only recalculate the upper bound for the binary search when an iteration saturates without satisfying $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$. In case of saturation one can for instance recalculate λ_2 iteratively by using the inverse iteration method, $v_2^{k+1} \propto (FF^T L_G FF^T - \lambda_2^{\mathrm{est}} FF^T D_G FF^T)^+ FF^T D_G FF^T v_2^k$, and normalizing such that $(v_2^{k+1})^T v_2^{k+1} = 1$.

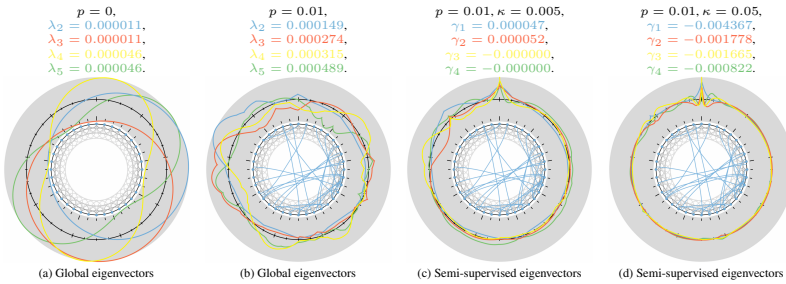
4 Illustrative Empirical Results

In this section, we will provide a detailed empirical evaluation of our method of semi-supervised eigenvectors and how they can be used for locally-biased machine learning. Our goal will be two-fold: first, to illustrate how the "knobs" of our method work; and second, to illustrate the usefulness of the method in a real application. To do so, we will consider:

- *Toy data.* In Section 4.1, we will consider one-dimensional examples of the popular "small world" model [22]. This is a parameterized family of models that interpolates between low-dimensional grids and random graphs; and, as such, it will allow us to illustrate the behavior of our method and it's various parameters in a controlled setting.
- *Handwritten image data.* In Section 4.2, we will consider the data from the MNIST digit data set [12]. These data have been widely-studied in machine learning and related areas and they have substantial "local heterogeneity"; and thus these data will allow us to illustrate how our method may be used to perform locally-biased versions of common machine learning tasks such as smoothing, clustering, and kernel construction.

4.1 Small-world Data

To illustrate how the "knobs" of our method work, and in particular how κ and γ interplay, we consider data constructed from the so-called small-world model. To demonstrate how semi-supervised eigenvectors can focus on specific target regions of a data graph to capture slowest modes of local variation, we plot semi-supervised eigenvectors around illustrations of (non-rewired and rewired) realizations of the small-world graph; see Figure 2.



(a) Global eigenvectors (b) Global eigenvectors (c) Semi-supervised eigenvectors (d) Semi-supervised eigenvectors

Figure 2: In each case, (a-d) the data consist of 3600 nodes, each connected to it's 8 nearest-neighbors. In the center of each subfigure, we show the nodes (blue) and edges (black and light gray are the local edges, and blue are the randomly-rewired edges). In each subfigure, we wrap a plot (black x-axis and gray background) visualizing the 4 smallest semi-supervised eigenvectors, allowing us to see the effect of random edges (different values of rewiring probability p) and degree of localization (different values of κ). Eigenvectors are color coded as blue, red, yellow, and green, starting with the one having the smallest eigenvalue. See the main text for more details.

In Figure 2.a, we show a graph with no randomly-rewired edges ($p = 0$) and a locality parameter κ such that the global eigenvectors are obtained. This yields a symmetric graph with eigenvectors corresponding to orthogonal sinusoids, *i.e.*, for all eigenvectors, except the all-ones with eigenvalue 0, the algebraic multiplicity is 2, *i.e.*, the first two capture the slowest mode of variation and correspond to a sine and cosine with equal random phase-shift (rotational ambiguity). In Figure 2.b, random edges have been added with probability $p = 0.01$ and the locality parameter κ is still chosen such that the global eigenvectors of the rewired graph are obtained. In particular, note small kinks in the eigenvectors at the location of the randomly added edges. Since the graph is no longer symmetric, all of the visualized eigenvectors have algebraic multiplicity 1. Moreover, note that the slow mode of variation in the interval on the top left; a normalized-cut based on the leading global eigenvector would extract this region since the remainder of the ring is more well-connected due to the degree of rewiring. In Figure 2.c, we see the same graph realization as in Figure 2.b, except that the semi-supervised eigenvectors have a seed node at the top of the circle and the correlation

parameter $\kappa_t = 0.005$. Note that, like the global eigenvectors, the local approach produces modes of increasing variation. In addition, note that the neighborhood around "11 o-clock" contains more mass, when compared with Figure 2.b; the reason for this is that this region is well-connected with the seed via a randomly added edge. Above the visualization we also show the γ_t that saturates κ_t, *i.e.*, γ_t is the Lagrange multiplier that defines the effective correlation κ_t. Not shown is that if we kept reducing κ, then γ_t would tend towards λ_{t+1}, and the respective semi-supervised eigenvector would tend towards the global eigenvector. Finally, in Figure 2.d, the desired correlation is increased to $\kappa = 0.05$ (thus decreasing the value of γ_t), making the different modes of variation more localized in the neighborhood of the seed. It should be clear that, in addition to being determined by the locality parameter, we can think of γ as a regularizer biasing the global eigenvectors towards the region near the seed set.

4.2 MNIST Digit Data

We now demonstrate the semi-supervised eigenvectors as a feature extraction preprocessing step in a machine learning setting. We consider the well-studied MNIST dataset containing 60000 training digits and 10000 test digits ranging from 0 to 9. We construct the complete 70000×70000 k-NN graph with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$, where σ_i^2 being the Euclidean distance to it's nearest neighbor, and we define the graph Laplacian in the usual way. We evaluate the semi-supervised eigenvectors in a transductive learning setting by disregarding the majority of labels in the entire training data. We then use a few samples from each class to seed our semi-supervised eigenvectors, and a few others to train a downstream classification algorithm. Here we choose to apply the SGT of [11] for two main reasons. First, the transductive classifier is inherently designed to work on a subset of global eigenvectors of the graph Laplacian, making it ideal for validating that our localized basis constructed by the semi-supervised eigenvectors can be more informative when we are solely interested in the "local heterogeneity" near a seed set. Second, using the SGT based on global eigenvectors is a good point of comparison, because we are only interested in the effect of our subspace representation. (If we used one type of classifier in the local setting, and another in the global, the classification accuracy that we measure would obviously be biased.) As in [11], we normalize the spectrum of both global and semi-supervised eigenvectors by replacing the eigenvalues with some monotonically increasing function. We use $\lambda_i = \frac{i^2}{k^2}$, *i.e.*, focusing on ranking among smallest cuts; see [5]. Furthermore, we fix the regularization parameter of the SGT to $c = 3200$, and for simplicity we fix $\gamma = 0$ for all semi-supervised eigenvectors, implicitly defining the effective $\kappa = [\kappa_1, \ldots, \kappa_k]^T$. Clearly, other correlation distributions and values of γ may yield subspaces with even better discriminative properties[1].

	#Semi-supervised eigenvectors for SGT						#Global eigenvectors for SGT					
Labeled points	1	2	4	6	8	10	1	5	10	15	20	25
1 : 1	0.39	0.39	0.38	0.38	0.38	0.36	0.50	0.48	0.36	0.27	0.27	0.19
1 : 10	0.30	0.31	0.25	0.23	0.19	0.15	0.49	0.36	0.09	0.08	0.06	0.06
5 : 50	0.12	0.15	0.09	0.08	0.07	0.06	0.49	0.09	0.08	0.07	0.05	0.04
10 : 100	0.09	0.10	0.07	0.06	0.05	0.05	0.49	0.08	0.07	0.06	0.04	0.04
50 : 500	0.03	0.03	0.03	0.03	0.03	0.03	0.49	0.10	0.07	0.06	0.04	0.04

Table 1: Classification error for the SGT based on respectively semi-supervised and global eigenvectors. The first column from the left encodes the configuration, *e.g.*, 1:10 interprets as 1 seed and 10 training samples from each class (total of 22 samples - for the global approach these are all used for training). When the seed is well determined and the number of training samples moderate (50:500) a single semi-supervised eigenvector is sufficient, where for less data we benefit from using multiple semi-supervised eigenvectors. All experiments have been repeated 10 times.

Here, we consider the task of discriminating between *fours* and *nines*, as these two classes tend to overlap more than other combinations. (A closed *four* usually resembles *nine* more than an "open" *four*.) Hence, we expect localization on low order global eigenvectors, meaning that class separation will not be evident in the leading global eigenvector, but instead will be "buried" further down the spectrum. Thus, this will illustrate how semi-supervised eigenvectors can represent relevant heterogeneities in a local subspace of low dimensionality. Table 1 summarizes our classification results based on respectively semi-supervised and global eigenvectors. Finally, Figure 3 and 4 illustrates two realizations for the 1:10 configuration, where the training samples are fixed, but where we vary

[1] A thorough analysis regarding the importance of this parameter will appear in the journal version.

the seed nodes, to demonstrate the influence of the seed. See the caption in these figures for further details.

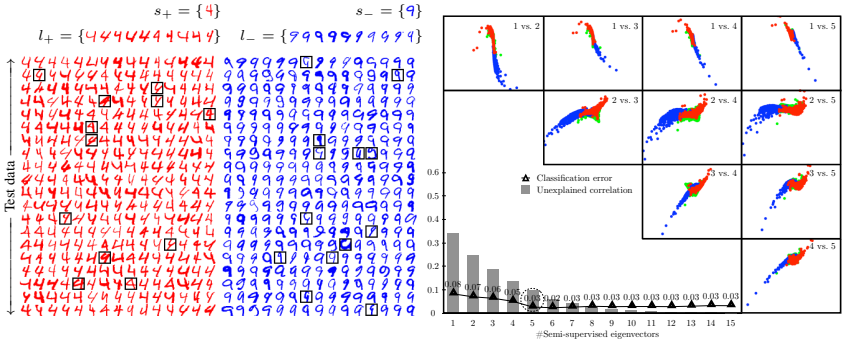


Figure 3: Left: Shows a subset of the classification results for the SGT based on 5 semi-supervised eigenvectors seeded in s_+ and s_-, and trained using samples l_+ and l_-. Misclassifications are marked with black frames. Right: Visualizes all test data spanned by the first 5 semi-supervised eigenvectors, by plotting each component as a function of the others. Red (blue) points correspond to 4 (9), whereas green points correspond to remaining digits. As the seed nodes are good representatives, we note that the eigenvectors provide a good class separation. We also plot the error as a function of local dimensionality, as well as the unexplained correlation, *i.e.*, initial components explain the majority of the correlation with the seed (effect of $\gamma = 0$). The particular realization based on the leading 5 semi-supervised eigenvectors yields an error of ≈ 0.03 (dashed circle).

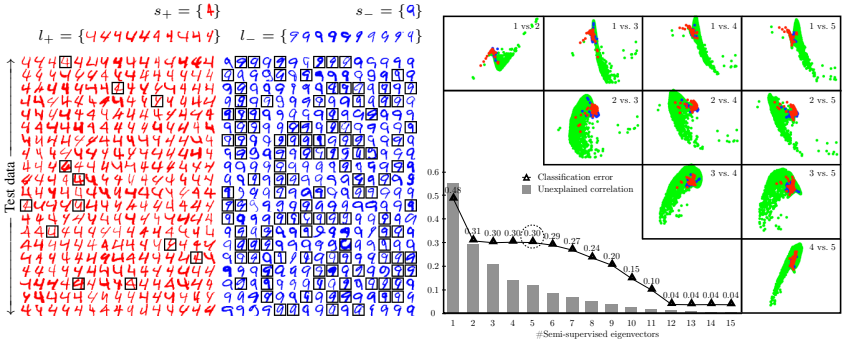


Figure 4: See the general description in Figure 3. Here we illustrate an instance where the s_+ shares many similarities with s_-, *i.e.*, s_+ is on the boundary of the two classes. This particular realization achieves a classification error of ≈ 0.30 (dashed circle). In this constellation we first discover localization on low order semi-supervised eigenvectors (≈ 12 eigenvectors), which is comparable to the error based on global eigenvectors (see Table 1), *i.e.*, further down the spectrum we recover from the bad seed and pickup the relevant mode of variation.

In summary: We introduced the concept of semi-supervised eigenvectors that are biased towards local regions of interest in a large data graph. We demonstrated the feasibility on a well-studied dataset and found that our approach leads to more compact subspace representations by extracting desired local heterogeneities. Moreover, the algorithm is scalable as the eigenvectors are computed by the solution to a sparse system of linear equations, preserving the low $\mathcal{O}(m)$ space complexity. Finally, we foresee that the approach will prove useful in a wide range of data analysis fields, due to the algorithm's speed, simplicity, and stability.

References

[1] R. Andersen, F.R.K. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 475–486, 2006.

[2] R. Andersen and K. Lang. Communities from seed sets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 223–232, 2006.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[5] O. Chapelle, J. Weston, and B. Schölkopf. Cluster Kernels for Semi-Supervised Learning. In Becker, editor, *NIPS 2002*, volume 15, pages 585–592, Cambridge, MA, USA, 2003.

[6] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition in data: Diffusion maps. *Proc. Natl. Acad. Sci. USA*, 102(21):7426–7431, 2005.

[7] A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Proceedings of th 11th International Conference on Computer Vision*, pages 1–8, 2007.

[8] W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114/115:815–839, 1989.

[9] T.H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

[10] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, 2003.

[11] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[12] Y. Lecun and C. Cortes. The MNIST database of handwritten digits.

[13] J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 695–704, 2008.

[14] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. Technical report, 2009. Preprint: arXiv:0912.0681.

[15] S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2057–2064, 2011.

[16] K.A. Norman, S.M. Polyn, G.J. Detre, and J.V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fmri data. *Trends in Cognitive Sciences*, 10(9):424–30, 2006.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[18] B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[19] D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proceedings of the 36th annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

[20] S.-H. Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, pages 2–14, 2010.

[21] S. Vigna. Spectral ranking. Technical report. Preprint: arXiv:0912.0238 (2009).

[22] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[23] S. X. Yu and J. Shi. Grouping with bias. In *Annual Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 1327–1334, 2002.

Appendix E

Information-based Kernel PCA Denoising by Semi-supervised Manifold Learning

---

Submitted to *Pattern Recognition Letters (PRL 2013)*

Information-based Kernel PCA Denoising by Semi-supervised Manifold Learning

Toke Jansen Hansen, Trine Julie Abrahamsen, Lars Kai Hansen
DTU Compute, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

Abstract

Kernel Principal Component Analysis (PCA) has proven a powerful tool for nonlinear feature extraction, and is often applied as a pre-processing step for classification algorithms. In denoising applications Kernel PCA provides the basis for dimensionality reduction, prior to the so-called *pre-image problem* where denoised feature space points are mapped back into input space. This problem is inherently ill-posed due to the non-bijective feature space mapping. We present a semi-supervised denoising scheme based on kernel PCA and the pre-image problem, where class labels on a subset of the data points are used to improve the denoising. Moreover, by warping the Reproducing Kernel Hilbert Space (RKHS) we also account for the intrinsic manifold structure yielding a Kernel PCA basis that also benefit from unlabeled data points. Our two main contributions are; 1) A generalization of Kernel PCA by incorporating a loss term, leading to an iterative algorithm for finding orthonormal components biased by the class labels, and 2) A fixed-point iteration for solving the pre-image problem based on a manifold warped RKHS. We prove viablity of the proposed methods on both

Email address: tjha@imm.dtu.dk, +45 45253888 (Toke Jansen Hansen)

synthetic data and images from The Amsterdam Library of Object Images (Geusebroek et al., 2005).

Keywords: Semi-supervised denoising, kernel PCA, pre-image problem

1. Introduction

In Principal Component Analysis (PCA) we seek an orthogonal basis that maximizes the explained variance of a data set. This basis can be found by computing eigenvectors of the centered covariance matrix, where the magnitude of an eigenvalue λ_i equals the amount of variance in the direction of the corresponding eigenvector $\boldsymbol{v}_i$, also denoted as the i^{th}*principal component*. In data compression, data is represented by a subset of the principal components having the largest eigenvalues, thereby ensuring that we retain as much variance as possible, whereas in denoising applications we deliberately drop directions with small variance (Mika et al., 1999).

When the data set contain nonlinear structures we cannot rely on linear PCA to provide a meaningful representation. Kernel PCA is the natural generalization of PCA, leveraging on the well known *kernel trick* to explain complicated nonlinear relations. We can think of the kernel PCA procedure as employing a function $\varphi : \mathcal{X} \mapsto \mathcal{H}$ that maps data from a $D_{\mathcal{X}}$-dimensional input space $\mathcal{X}$ to a $D_{\mathcal{H}}$-dimensional feature space $\mathcal{H}$ (possibly infinite dimensional), followed by performing linear PCA in $\mathcal{H}$. In practice we do never carry out the explicit mapping, but instead exploit the *kernel trick* stating that inner products in $\mathcal{H}$ can be computed in terms of kernel evaluations in $\mathcal{X}$, *i.e.*, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$. Hence, all algorithms that can be formulated solely in terms of inner products are applicable for the kernel trick, where the

function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ must fulfill Mercers condition, stating that $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ must be a positive definite matrix. A popular choice of kernel function is the Gaussian, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$, that has been successfully applied in both classification and denoising applications (Schölkopf et al., 1998).

For denoising purposes, we are interested in estimating the inverse mapping, φ^{-1}, known as the *pre-image problem*. For the Gaussian kernel the implicit mapping defined by φ is non-bijective leading to the inherently ill-posed pre-image problem. The fixed-point iteration described by Mika et al. (1999), provides an efficient scheme for determining the pre-images for Gaussian kernels, building upon standard gradient descent methods.

In this contribution we apply semi-supervised learning to construct a label informed kernel PCA basis. We achieve this, by extending the kernel PCA objective with a loss term and derive an efficient algorithm for computing an orthonormal basis biased towards a set of labeled training points. Furthermore, we derive a fixed-point iteration for finding an approximate pre-image for the kernel function introduced by Sindhwani et al. (2005). This Graph based kernel warps the corresponding RKHS to account for the manifold structure imposed by both labeled and unlabeled data points. The common goal for these two methods is to exploit labeled data to determine a more descriptive manifold representation. I.e., when using a fixed number of components we claim to achieve "better" denoised reconstructions than standard kernel PCA.

1.1. Related work

There is a vast literature on both kernel methods and semi-supervised learning, hence, for a general overview we refer to, e.g., Chapelle et al. (2006).

The pre-image problem was initially studied by Mika et al. (1999), who derived a fixed-point iteration for the Gaussian kernel. Bakir et al. (2004b) considered the pre-image problem for undirected graphs, and suggested a scheme for reconstructing graphs from the RKHS representation. Later studies considered regularization to make the pre-image problem more well behaved, see for instance Abrahamsen and Hansen (2011).

Walder et al. (2010) introduced the notion of semi-supervised kernel PCA by including a loss term, and derived solutions for objectives based on both squared and logistic losses. In particular, the squared loss can be interpreted as the Spectral Graph Transducer (SGT) by Joachims (2003), when the RKHS is defined by a graph based regularizer. In terms of the objective both Walder et al. (2010) and Joachims (2003) consider variations of a constrained eigenvalue problem and rely on a neat result by Gander et al. (1989) for a unique closed-form solution.

Another way of incorporating label information was introduced by Sindhwani et al. (2005) through the idea of warping the RKHS to account for the manifold structure imposed by both labeled and unlabeled data points, and derived the kernel

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{k}_x^\top (\boldsymbol{I} + \boldsymbol{L}\boldsymbol{K})^{-1} \boldsymbol{L} \boldsymbol{k}_y \tag{1}$$

where $\boldsymbol{k_y} = [k(\boldsymbol{y}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{y}, \boldsymbol{x}_N)]^\top$ and $\boldsymbol{L}$ is the combinatorial graph Laplacian, defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{K}$, where $\boldsymbol{D}$ is a diagonal matrix with $D_{ii} = \sum_{i=1}^{N} K_{ij}$. In the remainder of this paper, we will denote the above kernel function as the *Graph kernel*.

Our work can be considered extensions of Walder et al. (2010) and Joachims (2003), in that we generalize the objective with an orthogonality constraint

to enable the construction of more than one orthogonal basis vectors. Furthermore, we derive a fixed-point iteration for the pre-image problem based on the Graph kernel by Sindhwani et al. (2005), that directly relates to the SGT. However, we emphasize that generalizing the semi-supervised kernel PCA objective to allow for an arbitrary number orthogonal components is relevant, along the same line as extracting more than a single kernel PCA component when higher dimensional representations are needed to describe the signal manifold.

2. Methods

The remainder of this section will be outlined as follows. In Section 2.1 we extend the usual kernel PCA objective with a squared loss term, similar to the work of Walder et al. (2010), and develop a scheme for finding a semi-supervised kernel PCA basis of arbitrary dimensionality. In Section 2.2 we leverage on the ideas of Sindhwani et al. (2005) and apply them in the context of the pre-image problem, by deriving a fixed-point iteration for Graph kernel.

2.1. Semi-supervised kernel PCA

In semi-supervised kernel PCA we incorporate knowledge of the class labels on a subset of the data points. In this section we generalize the result of Walder et al. (2010) to account for multiple orthonormal components, thereby allowing us to compute a kernel PCA basis where the n^{th} direction is biased towards training labels with the constraint of being perpendicular to the previous $n-1$ components.

Figure 1 shows the original kernel PCA objective together with our modification that incorporates a least squares loss term in the form of an additional constraint. Note that we are explicit about the kernel PCA components being perpendicular in both the original objective and our modification, since in the latter case this constraint must be handled by an explicit projection onto the null space of previous components.

Kernel PCA

$$\max_{f_n \in \mathcal{H}} \quad \sum_{i=1}^{N} \left(f_n(\boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} f_n(\boldsymbol{x}_j) \right)^2$$

$$\text{s.t} \quad \|f_n\|_{\mathcal{H}}^2 = 1$$

$$\sum_{i=1}^{n-1} \langle f_n, f_i \rangle_{\mathcal{H}}^2 = 0$$

Semi-supervised kernel PCA

$$\max_{f_n \in \mathcal{H}} \quad \sum_{i=1}^{N} \left(f_n(\boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} f_n(\boldsymbol{x}_j) \right)^2$$

$$\text{s.t} \quad \|f_n\|_{\mathcal{H}}^2 = 1$$

$$\sum_{i=1}^{n-1} \langle f_n, f_i \rangle_{\mathcal{H}}^2 = 0$$

$$\sum_{i \in \mathcal{L}} (f_n(\boldsymbol{x}_i) - y_i)^2 \leq \omega$$

Figure 1: Left: The usual kernel PCA objective. Right: Our modified kernel PCA objective incorporating a least squares loss term. $\mathcal{L}$ is the set of labeled training data and ω determines the allowed derivation from the true labels.

For the original kernel PCA objective we can apply the representer theorem $f^\star(\cdot) = \sum_{i=1}^{N} \alpha_i^\star k(\boldsymbol{x}_i, \cdot)$ and form the derivative with respect to $\boldsymbol{\alpha}$ of the Lagrangian, leading to the following generalized eigenvalue problem

$$\boldsymbol{K}\boldsymbol{\alpha} = \lambda(\boldsymbol{K}^\top \boldsymbol{K} - \boldsymbol{K}^\top \boldsymbol{E}_N \boldsymbol{K})\boldsymbol{\alpha}, \tag{2}$$

where $\boldsymbol{E}_N$ is a matrix of size N with entries $\frac{1}{N}$.

To solve the extended semi-supervised objective efficiently we rewrite it in a similar manner as in Walder et al. (2010), where we minimize the norm

together with the squared loss term while keeping the variance fixed.

$$\underset{f_n \in \mathcal{H}}{\text{minimize}} \quad \|f_n\|_{\mathcal{H}}^2 + c \sum_{i \in \mathcal{L}} (f_n(\boldsymbol{x}_i) - y_i)^2 \tag{3}$$

$$\text{s.t} \quad \sum_{i=1}^{N} \left(f_n(\boldsymbol{x}_i) - \frac{1}{N} \sum_{j=1}^{N} f_n(\boldsymbol{x}_j) \right)^2 = s^2 \tag{4}$$

$$\sum_{i=1}^{n-1} \langle f_n, f_i \rangle_{\mathcal{H}}^2 = 0 \tag{5}$$

The main reason for the above formulation is that the linear part of the squared loss term makes the relationship between s and $f^\star$ non-trivial, but in this constellation we can control the relative importance of the respective terms via the parameters c and s^2. Applying the representer theorem yields

$$\underset{\boldsymbol{\alpha}_n \in \mathbb{R}^N}{\text{minimize}} \quad \boldsymbol{\alpha}_n^\top \boldsymbol{K} \boldsymbol{\alpha}_n + c \|\boldsymbol{K}_{\mathcal{L}} \boldsymbol{\alpha}_n - \boldsymbol{t}\|^2 \tag{6}$$

$$\text{s.t} \quad \boldsymbol{\alpha}_n^\top (\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K}) \boldsymbol{\alpha}_n = s^2 \tag{7}$$

$$\sum_{i=1}^{n-1} (\boldsymbol{\alpha}_n^\top \boldsymbol{K} \boldsymbol{\alpha}_i)^2 = 0 \tag{8}$$

where $\boldsymbol{t} \in \mathbb{R}^{|\mathcal{L}|}$ is a sub-vector of $\boldsymbol{y}$ that only takes indices $\mathcal{L}$, and likewise does $\boldsymbol{K}_{\mathcal{L}}$ denote the sub-matrix of $\boldsymbol{K}$ by taking rows $\mathcal{L}$. To account for the orthogonality constraint in Equation (8) we apply a projection operator on $\boldsymbol{\alpha}_n$, forcing the solution to be in the null space of previous solutions, see for instance Golub (1973). Let $\boldsymbol{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{n-1}]$ be the previous components, then the $\boldsymbol{S} = \text{Null}(\boldsymbol{K}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{K})$ is an orthonormal basis of size $N \times (N - n + 1)$ for the null space of $\boldsymbol{K}\boldsymbol{A}$ obtained from a singular value decomposition (SVD). Hence, by projectin $\boldsymbol{S}\boldsymbol{\alpha}_n$, the Lagrangian of the semi-supervised kernel PCA problem in Equation (6)-(8) can be formulated as a

$(N-n+1)$-dimensional problem

$$L = \boldsymbol{\alpha}_n^\top \boldsymbol{S}^\top \boldsymbol{K}\boldsymbol{S}\boldsymbol{\alpha}_n + c||\boldsymbol{K}_\mathcal{L}\boldsymbol{S}\boldsymbol{\alpha}_n - \boldsymbol{t}||^2 + \lambda(\boldsymbol{\alpha}_n^\top \boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}\boldsymbol{\alpha}_n - s^2) \tag{9}$$

Setting the partial derivatives to zero gives

$$\begin{aligned}\frac{\delta L}{\delta \boldsymbol{\alpha}_n} &= 2\boldsymbol{S}^\top \boldsymbol{K}\boldsymbol{S}\boldsymbol{\alpha}_n + 2c\boldsymbol{S}^\top \boldsymbol{K}_\mathcal{L}\boldsymbol{K}_\mathcal{L}\boldsymbol{S}\boldsymbol{\alpha}_n \\ &\quad - 2c\boldsymbol{S}^\top \boldsymbol{K}_\mathcal{L}\boldsymbol{t} + 2\lambda\boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}\boldsymbol{\alpha}_n \\ &= \boldsymbol{0} \end{aligned} \tag{10}$$

$$\frac{\delta L}{\delta \lambda} = \boldsymbol{\alpha}_n^\top \boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}\boldsymbol{\alpha}_n - s^2 = 0 \tag{11}$$

Leading to the following system of coupled equations

$$\begin{aligned}&\boldsymbol{S}^\top(\boldsymbol{K} + c\boldsymbol{K}_\mathcal{L}\boldsymbol{K}_\mathcal{L})\boldsymbol{S}\boldsymbol{\alpha}_n = \\ &-\lambda\boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}\boldsymbol{\alpha}_n + c\boldsymbol{S}^\top \boldsymbol{K}_\mathcal{L}\boldsymbol{t}\end{aligned} \tag{12}$$

$$\boldsymbol{\alpha}_n^\top \boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}\boldsymbol{\alpha}_n = s^2 \tag{13}$$

Substituting $\boldsymbol{C} = \boldsymbol{S}^\top(\boldsymbol{K} + c\boldsymbol{K}_\mathcal{L}\boldsymbol{K}_\mathcal{L})\boldsymbol{S}$, $\boldsymbol{b} = c\boldsymbol{S}^\top \boldsymbol{K}_\mathcal{L}\boldsymbol{t}$, and $\boldsymbol{P} = \boldsymbol{S}^\top(\boldsymbol{K}\boldsymbol{K} - \boldsymbol{K}\boldsymbol{E}_N\boldsymbol{K})\boldsymbol{S}$, these simplify to

$$\boldsymbol{C}\boldsymbol{\alpha}_n = -\lambda\boldsymbol{P}\boldsymbol{\alpha}_n + \boldsymbol{b} \tag{14}$$

$$\boldsymbol{\alpha}_n^\top \boldsymbol{P}\boldsymbol{\alpha}_n = s^2 \tag{15}$$

The first equation leads to

$$\boldsymbol{\alpha}_n = (\boldsymbol{C} + \lambda\boldsymbol{P})^{-1}\boldsymbol{b} \tag{16}$$

To saturate the second equation we can make use of the ideas in Gander et al. (1989), stating that λ should equal the smallest eigenvalue of the following generalized eigenvalue problem

$$\begin{pmatrix} \boldsymbol{C} & -\boldsymbol{P} \\ -\frac{1}{s^2}\boldsymbol{b}\boldsymbol{b}^\top & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\eta} \end{pmatrix} = \lambda \begin{pmatrix} \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P} \end{pmatrix} \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\eta} \end{pmatrix} \tag{17}$$

Typically this system is badly conditioned, so in practical applications we must solve this by other means. For $\lambda < \delta$ where δ is the smallest eigenvalue of the generalized eigenvalue problem $\boldsymbol{C}\boldsymbol{x} = \delta\boldsymbol{P}\boldsymbol{x}$, the solution will be unique if and only if the secular equation $\boldsymbol{\alpha}_n^\top \boldsymbol{P}\boldsymbol{\alpha}_n - s^2 = 0$ can be satisfied. Since the secular equation is strictly increasing for $\lambda \in]-\infty, \delta)$, we can instead perform a binary search in this range, in order to saturate $\boldsymbol{\alpha}_n^\top \boldsymbol{P}\boldsymbol{\alpha}_n = s^2$ with a sufficiently high precision. For more details we refer to Walder et al. (2010) and Gander et al. (1989).

2.2. The pre-image problem

Given a basis parameterized by a set of $\boldsymbol{\alpha}$'s determined by either standard kernel PCA or semi-supervised kernel PCA as described in the previous section, we are now interested in projecting a φ-mapped test point onto a principal subspace. For denoising applications we are interested in the projection onto the signal manifold, defined as a subspace of the RKHS spanned by the leading principal components. From the Representer Theorem, the projection of a feature space mapped test point onto the *n'th* principal component is

$$\beta_n(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_{ni} k_c(\boldsymbol{x}, \boldsymbol{x}_i) \tag{18}$$

where k_c is the centered kernel. The projection of $\varphi(\boldsymbol{x})$ onto the subspace spanned by the first q components will be denoted $P_q\varphi(\boldsymbol{x})$ and are given by

$$P_q\varphi(\boldsymbol{x}) = \sum_{n=1}^{q} \beta_n \sum_{i=1}^{N} \alpha_{ni}\varphi_c(\boldsymbol{x}_i) + \bar{\varphi} \tag{19}$$

where $\bar{\varphi} = \frac{1}{N}\sum_{i=1}^{N}\varphi(\boldsymbol{x}_i)$ is the mean of the φ-mapped data points and $\varphi_c(\boldsymbol{x}_i) = \varphi(\boldsymbol{x}_i) - \bar{\varphi}$ is the centered feature space mapping of $\boldsymbol{x}$ (Schölkopf et al., 1998).

For denoising purposes it is of interest to reconstruct a data point in input space that corresponds to a specific linearly denoised point in feature space, hence, applying the inverse map of φ. Thus, we are interested in finding a point $\boldsymbol{z} \in \mathcal{X}$ such that $\varphi(\boldsymbol{z}) = P_q\varphi(\boldsymbol{x})$ and we will call $\boldsymbol{z}$ the pre-image of $P_q\varphi(\boldsymbol{x})$.

The standard pre-image problem of reconstructing kernel PCA projections have been faced in a variety of ways, most of which are limited to a specific choice of kernel embedding (see e.g., Mika et al. (1999); Kwok and Tsang (2003); Dambreville et al. (2006); Bakir et al. (2004a)).

We follow the original work by Mika et al. (1999) and relax the problem to that of finding an approximate pre-image, i.e., a point in input space which maps into a point in feature space "as close as possible" to $P_q\varphi(\boldsymbol{x})$. To implement this search we seek to minimize the distance in the RKHS between $\varphi(\boldsymbol{z})$ and $P_q\varphi(\boldsymbol{x})$ with respect to $\boldsymbol{z}$. Thus, we use a quadratic objective function, which can be simplified as

$$\begin{aligned}\rho &= ||\varphi(\boldsymbol{z}) - P_q\varphi(\boldsymbol{x})||^2 \\ &= k(\boldsymbol{z},\boldsymbol{z}) - 2\sum_{n=1}^{N}\xi_n k(\boldsymbol{z},\boldsymbol{x}_n) + \Omega\end{aligned} \tag{20}$$

where all the $\boldsymbol{z}$-independent terms are collected in Ω, and $\xi_n = \tilde{\xi}_n + \frac{1}{N}(1 - \sum_{j=1}^{N} \tilde{\xi}_j)$, with $\tilde{\xi}_n = \sum_{i=1}^{q} \beta_i \alpha_{in}$.

In extrema, the derivative with respect to $\boldsymbol{z}$ is zero, which leads to the following fixed-point iteration for Gaussian kernels (Mika et al., 1999)

$$\begin{aligned} \boldsymbol{z}_{t+1} &= \frac{\sum_{n=1}^{N} \xi n \exp(-\gamma ||\boldsymbol{z}_t - \boldsymbol{x}_n||^2) \boldsymbol{x}_n}{\sum_{n=1}^{N} \xi n \exp(-\gamma ||\boldsymbol{z}_t - \boldsymbol{x}_n||^2)} \\ &= \frac{[\boldsymbol{\xi} \circ \boldsymbol{k}_{\boldsymbol{z}_t}]^\top \boldsymbol{X}}{\boldsymbol{\xi}^\top \boldsymbol{k}_{\boldsymbol{z}_t}} \end{aligned} \tag{21}$$

The cost in Equation (20) may be highly multi modal, leading to a nonlinear optimization problem, and hence the fixed-point iteration scheme can suffer from convergence to local minima. This typically implies sensitivity to the initial point $\boldsymbol{z}$ and can lead to instability of the denoising solution, see e.g., Abrahamsen and Hansen (2011).

Similarly, we now seek a fixed-point iteration for determining the pre-image when using the Graph kernel, $\tilde{k}$, as defined in Equation (1). When updating the pre-image estimate we will for simplicity assume that the pre-image itself is not part of $\boldsymbol{K}$. Thereby, we avoid the inversion of $(\boldsymbol{I} + \boldsymbol{L}\boldsymbol{K})^{-1}$ at every iteration that scales cubically. The effects of this relaxation will be reduced if the manifold is well defined by the training samples. By letting $\boldsymbol{M} := (\boldsymbol{I} + \boldsymbol{L}\boldsymbol{K})^{-1}\boldsymbol{L}$, the Graph kernel simplifies to $\tilde{k}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{k}_x^\top \boldsymbol{M} \boldsymbol{k}_y$.

We now expand the $\boldsymbol{z}$ dependent terms of the cost function in Eq. (20)

for this kernel

$$\rho = \tilde{k}(\boldsymbol{z}, \boldsymbol{z}) - 2\sum_{n=1}^{N} \xi_n \tilde{k}(\boldsymbol{z}, \boldsymbol{s}_n) \tag{22}$$

$$\begin{aligned}
&= \exp(-\gamma||\boldsymbol{z} - \boldsymbol{z}||^2) \\
&- \sum_{i,j=1}^{N} \exp(-\gamma||\boldsymbol{z} - \boldsymbol{s}_i||^2) M_{ij} \exp(-\gamma||\boldsymbol{z} - \boldsymbol{s}_j||^2) \\
&- 2\Big[\sum_{n=1}^{N} \xi_n \big[\exp(-\gamma||\boldsymbol{z} - \boldsymbol{s}_n||^2) - \\
&\sum_{i,j=1}^{N} \exp(-\gamma||\boldsymbol{z} - \boldsymbol{s}_i||^2) M_{ij} \exp(-\gamma||\boldsymbol{s}_n - \boldsymbol{s}_j||^2)\big]\Big]
\end{aligned} \tag{23}$$

Again the minima of Equation (23) are among points in which the derivative with respect to $\boldsymbol{z}$ is zero

$$\begin{aligned}
&\sum_{i,j=1}^{N} \Big[M_{ij}(2\boldsymbol{z} - \boldsymbol{s}_i - \boldsymbol{s}_j) \exp(-\gamma(||\boldsymbol{z} - \boldsymbol{s}_i||^2 + ||\boldsymbol{z} - \boldsymbol{s}_j||^2)) \\
&- \Big[\sum_{n=1}^{N} \xi_n \big[-2(\boldsymbol{z} - \boldsymbol{s}_n) \exp(-\gamma||\boldsymbol{z} - \boldsymbol{s}_n||^2) \\
&+ \sum_{i,j=1}^{N} 2M_{ij}(\boldsymbol{z} - \boldsymbol{s}_i + \boldsymbol{s}_n - \boldsymbol{s}_j) \cdot \\
&\exp(-\gamma(||\boldsymbol{z} - \boldsymbol{s}_i||^2 + ||\boldsymbol{s}_n - \boldsymbol{s}_j||^2))\big]\Big]\Big] = \boldsymbol{0}
\end{aligned} \tag{24}$$

Hence, we arrive at the following fixed-point iteration scheme for the Graph kernel

$$\begin{aligned}
\boldsymbol{z}_{t+1} = &\frac{\left[(\boldsymbol{M} \circ (\boldsymbol{k}_{\boldsymbol{z}_t}\boldsymbol{k}_{\boldsymbol{z}_t}^\top - \boldsymbol{k}_{\boldsymbol{z}_t}(\boldsymbol{K}\boldsymbol{\xi})^\top - (\boldsymbol{K}\boldsymbol{\xi})\boldsymbol{k}_{\boldsymbol{z}_t}^\top))\boldsymbol{1}\right]^\top \boldsymbol{X}}{(\boldsymbol{k}_{\boldsymbol{z}_t}^\top \boldsymbol{M} + \boldsymbol{\xi}^\top - 2\boldsymbol{\xi}^\top \boldsymbol{K}\boldsymbol{M})\boldsymbol{k}_{\boldsymbol{z}_t}} \\
&+ \frac{[\boldsymbol{M} \circ \boldsymbol{\xi} \circ \boldsymbol{k}_{\boldsymbol{z}_t}]^\top \boldsymbol{X}}{(\boldsymbol{k}_{\boldsymbol{z}_t}^\top \boldsymbol{M} + \boldsymbol{\xi}^\top - 2\boldsymbol{\xi}^\top \boldsymbol{K}\boldsymbol{M})\boldsymbol{k}_{\boldsymbol{z}_t}}
\end{aligned} \tag{25}$$

3. Experimental results

In the following we evaluate the performance of denoising by semi-supervised kernel PCA on two data sets. To get some insights on the properties of the proposed methods we design a two-dimensional two-class problem with non-linear-separable manifolds by two intertwined spirals. Furthermore, we test the performance on a subset of The Amsterdam Library of Object Images (ALOI) database of images (Geusebroek et al., 2005).

3.1. Simulated data

To investigate the denoising performance of the proposed methods we construct a simple two class two dimensional synthetic data set as shown in the left panel of Figure 2. The data consists of two noisy entangled spirals where a random subset of the observations have label information.

For the experiments we retain 3 principal components and measure the quality of the denoising scheme by the mean squared error (MSE) of the reconstruction of a test set. Initially, we investigate the performance for varying signal-to-noise ratios by adding i.i.d. Gaussian noise to the data with zero mean and variance, σ^2. For all experiments we fixed the parameters of the semi-supervised model to $s = 10$ and $c = 2$.

We use 300 observations for training of which the label is known for 50 randomly chosen points from each class. The test set contains 100 unlabeled test points. The kernel-parameter is fixed to $\gamma = 5$. The results are summarized in the right panel of Figure 2, where we show error bars on the MSE as a function of the standard deviation of the Gaussian noise. It is evident that the semi-supervised reconstructions outperform their unsuper-

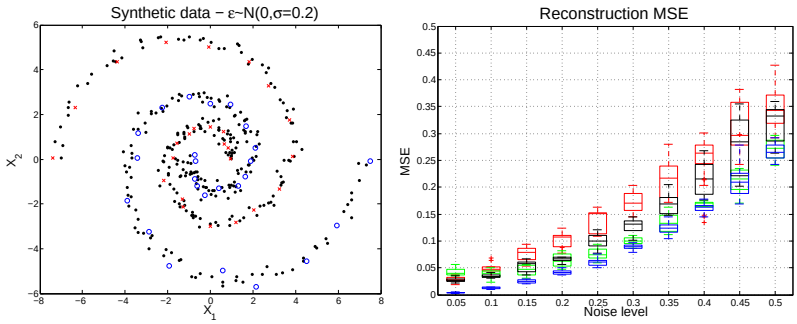


Figure 2: The left panel ilustrates the synthetic data used for our experiments. Colored samples are labeled whereas black samples are unlabeled. The right panel shows the mean squared error as a function of the standard deviation of the added Gaussian noise, σ. The green bars are kernel PCA with the Graph kernel, while the blue bars are semi-supervised kernel PCA with Graph kernel. Similarly, the red bars are kernel PCA with the Gaussian kernel, while the black bars are semi-supervised kernel PCA with the Gaussian kernel. Incorporating label information by the Graph kernel is seen to outperform the Gaussian kernel, and for both choices of kernel, semi-supervised learning is found to improve performance.

vised counterparts for both the Gaussian and the Graph kernel for all noise levels. Furthermore, using the Graph kernel clearly leads to a better reconstruction measured by a lower MSE indicating a more descriptive manifold representation.

In order to investigate how much we learn from the unlabeled versus labeled samples we generate learning curves by fixing the noise level to $\sigma = 0.15$ and vary the number of observations used to learn the manifold structure. We learn the manifold fully supervised (with all labels known), unsupervised (standard kernel PCA), and semi-supervised (1/3 of the labels known) and

compare the MSE of the pre-image reconstructions. The results are summarized in Figure 3, where the left panel shows the results achieved using the Gaussian kernel, and the right panel shows the results using the Graph kernel. As expected, for a fixed training set size fully supervised learning is preferable while completely unsupervised learning performs the worst. This tendency is less clear when only few samples are available. For the Graph kernel having 1/3 of the labels yields results comparable to knowing all labels. This is due to correct label propagation since the manifold assumption holds. It is important to notice how adding unlabeled samples significantly lower the MSE for both kernels. This is evident by comparing, e.g., the MSE for the supervised methods for $N = 100$ with the MSE achieved using the semi-supervised scheme for $N = 300$ (i.e, 100 labeled samples and 200 unlabeled). For both kernel functions adding unlabeled observations leads to a significant lower MSE.

3.2. Amsterdam Library of Object Images

The Amsterdam Library of Object Images is a collection of images of 1000 objects that have been recorded for scientific purposes (Geusebroek et al., 2005). We consider a subset of 15 objects from the view point data set from this collection, where the view point is shifted in steps of 5° yielding a total of 72 images of each object. We treat each object as a class and assume to have 5 labeled samples (and 67 unlabeled) per class. Due to space constraints, we limit this section to a comparison between the unsupervised Gaussian kernel PCA approach and the semi-supervised Graph kernel PCA approach, as the two other combinations have shown to fall in-between the performance of the these two methodologies.

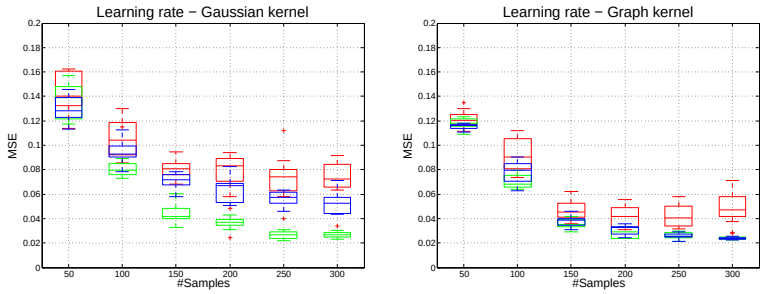


Figure 3: Illustrates the learning curves (mean squared error as a function of the training set size) for the synthetic data set depicted in Figure 2. The left panel shows the learning rates for the Gaussian kernel, while the right panel shows the learning rates for Graph kernel. The red bars are usual kernel PCA, i.e., unsupervised. Green bars: supervised kernel PCA, i.e., all training samples are labeled. Blue bars: semi-supervised kernel PCA with 1/3 of the training samples are labeled.

We construct a denoising problem by randomly adding two images from the database. The intensity of one of the images will be half of the intensity of the other, and the goal is to reconstruct the dominant image. For the semi-supervised methods we assume that the class label of the test image is known. Knowing the label of the test sample is justified by the fact that we are not focusing on classification, but merely aiming to incorporate side information for improved denoising. In case the test label was unknown an initial classification step can be performed using the semi-supervised kernel PCA basis, since each leading eigenvector can be interpreted as an one-vs-rest classifier.

Figure 4 shows examples of the results using our proposed methods. For all experiments, we retain $q = 10$ PC's and for the semi-supervised model

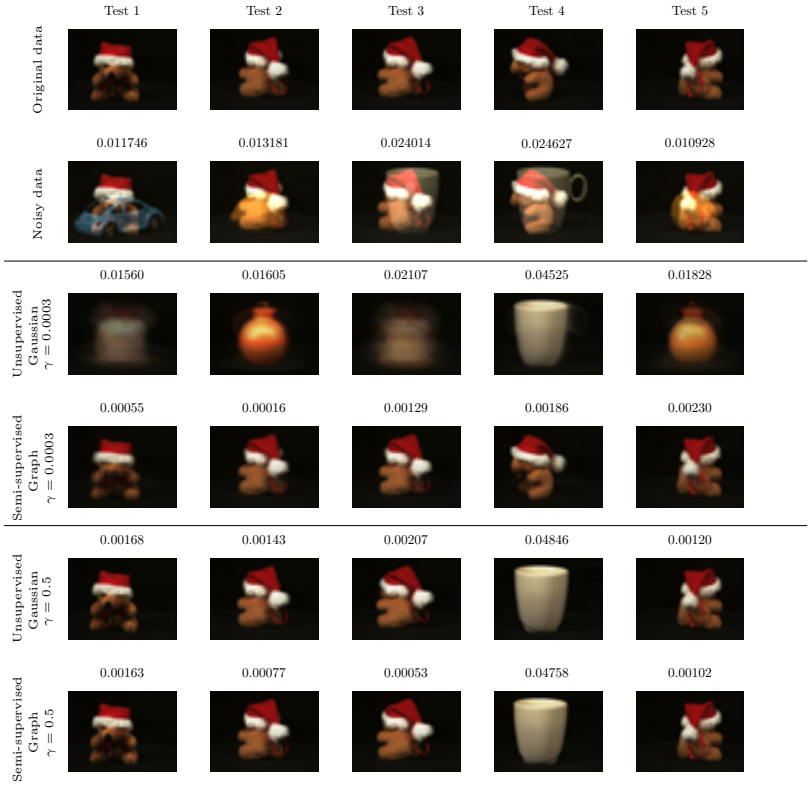


Figure 4: Examples of denoised images form the ALOI database. The first row shows the original test images that we seek to reconstruct, while the second row shows the constructed noisy test images. Above each image in row 2-6 are shown the MSE with respect to the original test image. The third and fourth row shows denoised images based on respectively unsupervised Gaussian and semi-supervised Graph kernel PCA; for both methods the kernel width has been fixed to $\gamma = 0.0204$ (leftmost point in Figure 5), and both utilize $q = 10$ PC's for the denosing task. The final two rows shows similar results but these are based on a more nonlinear kernel, where the parameter was fixed to $\gamma = 0.5$.

we set $s = 10$ and $c = 2$ as for the synthetic data. The top panel shows the original dominant test image, whereas the second panel shows the artificially constructed "noisy" image. The remaining panels show the denoised reconstructions for two choices of γ using unsupervised Gaussian kernel PCA and semi-supervised Graph kernel PCA respectively. The MSE with respect to the original test image is given above each image. It is evident both visually and from the MSE that the semi-supervised approach yields better results in all cases and it is more robust to the choice of kernel width, and aligns better with the original image (note the slight rotations of the reconstructions). For the most linear choice of kernel as measured by the kernel width, using the unsupervised Gaussian kernel is seen to fail in all cases while the semi-supervised version reconstruct meaningful images. For the more non-linear embedding the two reconstructions visually appear to be similar, however in terms of the MSE the semi-supervised kernel PCA is found to still be slightly superior.

Figure 5 shows the MSE as a function of the non-linearity of the kernel embedding. We find that a nearest neighbor reconstruction (large γ) is close to optimal measured by the MSE as seen in Figure 5. This can be explained by the fine angular sampling within each class and by the complex nature of the signal manifold relative to the low number of PC's retained. However, we emphasize that the MSE for the Graph kernel has an significantly lower envelope than the other method, and that good results are obtained for a much broader range of kernel width parameters, making the Graph kernel easier to deploy in practice.

For the unsupervised Gaussian kernel, the poor performance in the linear

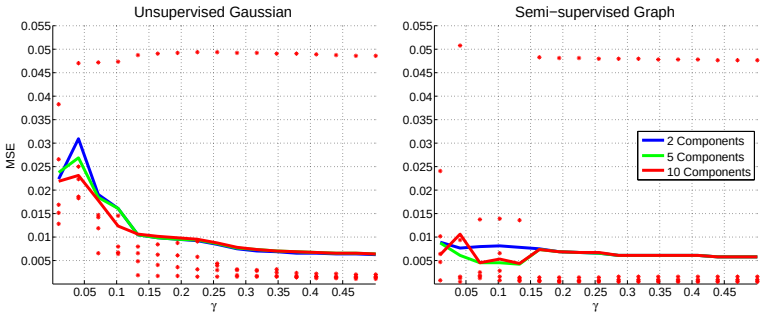


Figure 5: Shows the mean squared error as a function of the kernel width using a varying number of PCs to describe the manifold. Each point correspond to the MSE of one of the images in Figure 4 while the solid lines are the mean across all images. By comparing the right and left plots, it can be seen that the semi-supervised approach leads to better performance across all values of γ and in particular in the very linear regime. For both methods it is evident that the MSE is minimal for very large values of γ and hence very non-linear kernel embeddings, thereby suggesting that the optimal reconstruction is close a 1-nearest-neighbor approach.

regime can be explained as the consequence of the recovered leading principal components being unrelated to the denoising task of interest, thereby resulting in a high MSE. However, it should be noted that the task related components are still "hidden" in the span of the eigenvectors of the kernel matrix as long as it is not rank deficient. The key to the success of the semi-supervised approach is that the labeled samples forces the leading principal components to align with the task relevant directions independently of the choice γ. Thereby making this approach much less sensitive to the kernel width.

4. Conclusions

We have proposed two variants for incorporating label information into kernel PCA denoising. By extending the work of Walder et al. (2010) we derived an iterative scheme for finding more than one basis vector, leading to a semi-supervised kernel PCA framework that extends to a multidimensional orthonormal basis biased towards the labeled data.

Additionally, we derived a fixed-point iteration for the pre-image problem for the Graph kernel introduced by Sindhwani et al. (2005) as another way of including label information in the kernel PCA denoising scheme.

Viability was proven on both simulated data and images from the ALOI database. The experiments validated that semi-supervised learning can yield a more descriptive representation of the signal manifold in kernel PCA, and thereby improve the denoising performance compared to classical unsupervised kernel PCA denoising.

References

Abrahamsen, T. J., Hansen, L. K., 2011. Regularized pre-image estimation for kernel pca de-noising: Input space regularization and sparse reconstruction. Journal of Signal Processing Systems.

Bakir, G. H., Weston, J., Schölkopf, B., 2004a. Learning to find pre-images. In: Advances in Neural Information Processing Systems 16. MIT Press, pp. 449–456.

Bakir, G. H., Zien, A., Tsuda, K., 2004b. Learning to find graph pre-images.

In: Pattern Recognition. Vol. 3175 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 253–261.

Chapelle, O., Schölkopf, B., Zien, A. (Eds.), 2006. Semi-Supervised Learning. MIT Press.

Dambreville, S., Rathi, Y., Tannenbaum, A., 2006. Statistical shape analysis using kernel PCA. In: IS&T/SPIE Symposium on Electrical Imaging.

Gander, W., Golub, G. H., von Matt, U., 1989. A constrained eigenvalue problem. Linear Algebra and its Applications 114(115), 815 – 839.

Geusebroek, J. M., Burghouts, G. J., Smeulders, A. W. M., 2005. The amsterdam library of object images. International Journal of Computer Vision 61 (1), 103–112.

Golub, G. H., 1973. Some modified matrix eigenvalue problems. SIAM Review 15 (2), 318–334.

Joachims, T., 2003. Transductive learning via spectral graph partitioning. In: Proceedings of the International Conference on Machine Learning. pp. 290–297.

Kwok, J. T., Tsang, I. W., 2003. The pre-image problem in kernel methods. In: Proceedings of the International Conference on Machine Learning. pp. 408–415.

Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., Rätsch, G., 1999. Kernel pca and de-noising in feature spaces. In: Advances in neural information processing systems II. MIT Press, pp. 536–542.

281 Schölkopf, B., Smola, A., Müller, K.-R., 1998. Nonlinear component analysis
282 as a kernel eigenvalue problem. Neural Computation 10 (5), 1299–1319.

283 Sindhwani, V., Niyogi, P., Belkin, M., 2005. Beyond the point cloud: from
284 transductive to semi-supervised learning. In: Proceedings of the 22nd in-
285 ternational conference on Machine learning. ACM, pp. 824–831.

286 Walder, C., Henao, R., Mørup, M., Hansen, L. K., 2010. Semi-supervised
287 kernel pca. CoRR abs/1008.1398.

Appendix F

Semi-supervised Eigenvectors for Large-scale Locally-biased Learning

---

Submitted to *Journal of Machine Learning Research (JMLR 2013)*

Semi-supervised Eigenvectors for Large-scale Locally-biased Learning*

Toke J. Hansen † Michael W. Mahoney ‡

Abstract

In many applications, one has side information, *e.g.*, labels that are provided in a semi-supervised manner, about a specific target region of a large data set, and one wants to perform machine learning and data analysis tasks "nearby" that prespecified target region. For example, one might be interested in the clustering structure of a data graph near a prespecified "seed set" of nodes, or one might be interested in finding partitions in an image that are near a prespecified "ground truth" set of pixels. Locally-biased problems of this sort are particularly challenging for popular eigenvector-based machine learning and data analysis tools. At root, the reason is that eigenvectors are inherently global quantities, thus limiting the applicability of eigenvector-based methods in situations where one is interested in very local properties of the data.

In this paper, we address this issue by providing a methodology to construct *semi-supervised eigenvectors* of a graph Laplacian, and we illustrate how these locally-biased eigenvectors can be used to perform *locally-biased machine learning*. These semi-supervised eigenvectors capture successively-orthogonalized directions of maximum variance, conditioned on being well-correlated with an input seed set of nodes that is assumed to be provided in a semi-supervised manner. We show that these semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations; and we also describe several variants of our basic method that have improved scaling properties. We provide several empirical examples demonstrating how these semi-supervised eigenvectors can be used to perform locally-biased learning; and we discuss the relationship between our results and recent machine learning algorithms that use global eigenvectors of the graph Laplacian.

Keywords: Semi-supervised eigenvectors, spectral, local, kernel, machine learning

1 Introduction

In many applications, one has information about a specific target region of a large data set, and one wants to perform common machine learning and data analysis tasks "nearby" the pre-specified target region. In such situations, eigenvector-based methods such as those that have been popular in machine learning in recent years tend to have serious difficulties. At root, the reason is that eigenvectors, *e.g.*, of Laplacian matrices of data graphs, are inherently *global* quantities, and thus they might not be sensitive to very *local* information. Motivated by this, we consider the problem of finding a set of locally-biased vectors—we will call them *semi-supervised eigenvectors*—that inherit many of the "nice" properties that the leading nontrivial global eigenvectors of a graph Laplacian

*A preliminary version of parts of this paper appeared in the Proceedings of the 2012 NIPS Conference [24].

†Department of Applied Mathematics and Computer Science, Technical University of Denmark, tjha@imm.dtu.dk.

‡Department of Mathematics, Stanford University. mmahoney@cs.stanford.edu.

have—for example, that capture "slowly varying" modes in the data, that are fairly-efficiently computable, that can be used for common machine learning and data analysis tasks such as kernel-based and semi-supervised learning, etc.—so that we can perform what we will call *locally-biased machine learning* in a principled manner.

1.1 Locally-biased Learning

By *locally-biased machine learning*, we mean that we have a data set, *e.g.*, represented as a graph, and that we have information, *e.g.*, given in a semi-supervised manner, that certain "regions" of the data graph are of particular interest. In this case, we may want to focus predominantly on those regions and perform data analysis and machine learning, *e.g.*, classification, clustering, ranking, etc., that is "biased toward" those pre-specified regions. Examples of this include the following.

- *Locally-biased community identification.* In social and information network analysis, one might have a small "seed set" of nodes that belong to a cluster or community of interest [2, 35]; in this case, one might want to perform link or edge prediction, or one might want to "refine" the seed set in order to find other nearby members.
- *Locally-biased image segmentation.* In computer vision, one might have a large corpus of images along with a "ground truth" set of pixels as provided by a face detection algorithm [18, 37, 38]; in this case, one might want to segment entire heads from the background for all the images in the corpus in an automated manner.
- *Locally-biased neural connectivity analysis.* In functional magnetic resonance imaging applications, one might have small sets of neurons that "fire" in response to some external experimental stimulus [42]; in this case, one might want to analyze the subsequent temporal dynamics of stimulation of neurons that are "nearby," either in terms of connectivity topology or functional response, members of the original set.

In each of these examples, the data are modeled by a graph—which is either "given" from the application domain or is "constructed" from feature vectors obtained from the application domain—and one has information that can be viewed as semi-supervised in the sense that it consists of exogeneously-specified "labels" for the nodes of the graph. In addition, there are typically a relatively-small number of labels and one is interested in obtaining insight about the data graph nearby those labels.

These examples present considerable challenges for standard global spectral techniques and other traditional eigenvector-based methods. (Such eigenvector-based methods have received attention in a wide range of machine learning and data analysis applications in recent years. They have been useful, for example, in non-linear dimensionality reduction [3, 13]; in kernel-based machine learning [53]; in Nyström-based learning methods [65, 58]; spectral partitioning [50, 54, 41], and so on.) At root, the reason is that eigenvectors are inherently global quantities, thus limiting their applicability in situations where one is interested in very local properties of the data. That is, very local information can be "washed out" and essentially invisible to these globally-optimal vectors. For example, a sparse cut in a graph may be poorly correlated with the second eigenvector and thus invisible to a method based only on eigenvector analysis. Similarly, if one has semi-supervised information about a specific target region in the graph, as in the above examples, one might be interested in finding clusters near this prespecified local region in a semi-supervised manner; but this

local region might be essentially invisible to a method that uses only global eigenvectors. Finally, one might be interested in using kernel-based methods to find "local correlations" or to regularize with respect to a "local dimensionality" in the data, but this local information might be destroyed in the process of constructing kernels with traditional kernel-based methods.

1.2 Semi-supervised Eigenvectors

In this paper, we provide a methodology to construct what we will call *semi-supervised eigenvectors* of a graph Laplacian; and we illustrate how these locally-biased eigenvectors (locally-biased in the sense that they will be well-correlated with the input seed set of nodes or that most of their "mass" will be on nodes that are "near" that seed set) inherit many of the properties that make the leading nontrivial global eigenvectors of the graph Laplacian so useful in applications. In order to make this method useful, there should ideally be a "knob" that allows us to interpolate between very local and the usual global eigenvectors, depending on the application at hand; we should be able to use these vectors in common machine learning pipelines to perform common machine learning tasks; and the intuitions that make the leading k nontrivial global eigenvectors of the graph Laplacian useful should, to the extent possible, extend to the locally-biased setting. To achieve this, we will formulate an optimization ansatz that is a variant of the usual global spectral graph partitioning optimization problem that includes a natural locality constraint as well as an orthogonality constraint, and we will iteratively solve this problem.

In more detail, assume that we are given as input a (possibly weighted) data graph $G = (V, E)$, an indicator vector s of a small "seed set" of nodes, a *correlation parameter* $\kappa \in [0, 1]$, and a positive integer k. Then, informally, we would like to construct k vectors that satisfy the following bicriteria: first, each of these k vectors is well-correlated with the input seed set; and second, those k vectors describe successively-orthogonalized directions of maximum variance, in a manner analogous to the leading k nontrivial global eigenvectors of the graph Laplacian. (We emphasize that the seed set s of nodes, the integer k, and the correlation parameter κ are part of the input; and thus they should be thought of as being available in a semi-supervised manner.) Somewhat more formally, our main algorithm, Algorithm 1 in Section 3, returns as output k semi-supervised eigenvectors; each of these is the solution to an optimization problem of the form of Generalized LocalSpectral in Figure 1, and thus each "captures" (say) κ/k of the correlation with the seed set. Our main theoretical result, described in Section 3, states that these vectors define successively-orthogonalized directions of maximum variance, conditioned on being κ/k-well-correlated with an input seed set s; and that each of these k semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations. To extend the practical applicability of this basic result, we will in Section 4 describe several heuristic extensions of our basic framework that will make it easier to apply the method of semi-supervised eigenvectors at larger size scales. These extensions involve using the so-called Nyström method, computing one locally-biased eigenvector and iteratively "peelling off" successive components of interest, as well as performing random walks that are "local" in a stronger sense than our basic method considers.

Finally, in order to illustrate how the method of semi-supervised eigenvectors performs in practice, we also provide a detailed empirical evaluation using a wide range of both small-scale as well as larger-scale data. In particular, we consider two small data sets, one consisting of graphs generated from a popular network generation model and the other data drawn from Congressional roll call voting patterns, in order to illustrate the basic method; we consider graphs constructed from the widely-studied MNIST digit data, in order to illustrate how the method performs on a data set

that is widely-known in the machine learning community; and we consider two larger data sets, one consisting of Internet graphs and the other consisting of graphs constructed from fMRI medical imaging, in order to illustrate how the method performs in larger-scale applications.

1.3 Related Work

From a technical perspective, the work most closely related to ours is the recently-developed "local spectral method" of Mahoney *et al.* [37]. The original algorithm of Mahoney *et al.* [37] introduced a methodology to construct a locally-biased version of the *leading* nontrivial eigenvector of a graph Laplacian and also showed (theoretically and empirically in a social network analysis application) that that the resulting vector could be used to partition a graph in a locally-biased manner. From this perspective, our extension incorporates a natural orthogonality constraint that successive vectors need to be orthogonal to previous vectors. Subsequent to the work of [37], [38] applied the algorithm of [37] to to the problem of finding locally-biased cuts in a computer vision application. Similar ideas have also been applied somewhat differently. For example, [2] use locally-biased random walks, *e.g.*, short random walks starting from a small seed set of nodes, to find clusters and communities in graphs arising in Internet advertising applications; [35] used locally-biased random walks to characterize the local and global clustering structure of a wide range of social and information networks; and [29] developed the Spectral Graph Transducer, which performs transductive learning via spectral graph partitioning.

The objectives in both [29] and [37] are constrained eigenvalue problems that can be solved by finding the smallest eigenvalue of an asymmetric generalized eigenvalue problem; but in practice this procedure can be highly unstable [20]. The algorithm of [29] reduces the instabilities by performing all calculations in a subspace spanned by the d smallest eigenvectors of the graph Laplacian; whereas the algorithm of [37] performs a binary search, exploiting the monotonic relationship between a control parameter and the corresponding Lagrange multiplier. The form of our optimization problem also has similarities to other work in computer vision applications: *e.g.*, [66] and [18] find good conductance clusters subject to a set of linear constraints.

In parallel, [3] and a large body of subsequent work including [13] used (the usual global) eigenvectors of the graph Laplacian to perform dimensionality reduction and data representation, in unsupervised and semi-supervised settings [59, 51, 67]. Typically, these methods construct some sort of local neighborhood structure around each data point, and they optimize some sort of global objective function to go "from local to global" [52]. In some cases, these methods can be understood in terms of data drawn from an hypothesized manifold [5], and more generally they have proven useful for denoising and learning in semi-supervised settings [4, 6]. These methods are based on spectral graph theory [12]; and thus many of these methods have a natural interpretation in terms of diffusions and kernel-based learning [53, 31, 57, 11, 23]. These interpretations are important for the usefulness of these global eigenvector methods in a wide range of applications. As we will see, many (but not all) of these interpretations can be ported to the "local" setting, an observation that was made previously in a different context [14].

Many of these diffusion-based spectral methods also have a natural interpretation in terms of spectral ranking [61]. "Topic sensitive" and "personalized" versions of these spectral ranking methods have also been studied [26, 28]; and these were the motivation for diffusion-based methods to find locally-biased clusters in large graphs [55, 1, 37]. Our optimization ansatz is a generalization of the linear equation formulation of the PageRank procedure [43, 37, 61]; and its solution involves Laplacian-based linear equation solving, which has been suggested as a primitive is of more general

interest in large-scale data analysis [60].

1.4 Outline of the Paper

In the next section, Section 2, we will provide notation and some background and discuss related work. Then, in Section 3 we will present our main algorithm and our main theoretical result justifying the algorithm; and in Section 4 we will present several extensions of our basic method that will help for certain larger-scale applications of the method of semi-supervised eigenvectors. In Section 5, we present an empirical analysis, including both toy data to illustrate how the "knobs" of our method work, as well as applications to realistic machine learning and data analysis problems; and in Section 6 we present a brief discussion and conclustion.

2 Background and Notation

Let $G = (V, E, w)$ be a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges, in which edge $\{i, j\}$ has weight w_{ij}. For a set of vertices $S \subseteq V$ in a graph, the *volume of* S is $\mathrm{vol}(S) \stackrel{\mathrm{def}}{=} \sum_{i \in S} d_i$, in which case the *volume of the graph* G is $\mathrm{vol}(G) \stackrel{\mathrm{def}}{=} \mathrm{vol}(V) = 2m$. In the following, $A_G \in \mathbb{R}^{V \times V}$ will denote the adjacency matrix of G, while $D_G \in \mathbb{R}^{V \times V}$ will denote the diagonal degree matrix of G, *i.e.*, $D_G(i,i) = d_i = \sum_{\{i,j\} \in E} w_{ij}$, the weighted degree of vertex i. The Laplacian of G is defined as $L_G \stackrel{\mathrm{def}}{=} D_G - A_G$. (This is also called the combinatorial Laplacian, in which case the normalized Laplacian of G is $\mathcal{L}_G \stackrel{\mathrm{def}}{=} D_G^{-1/2} L_G D_G^{-1/2}$.)

The Laplacian is the symmetric matrix having quadratic form $x^T L_G x = \sum_{ij \in E} w_{ij}(x_i - x_j)^2$, for $x \in \mathbb{R}^V$. This implies that L_G is positive semidefinite and that the all-one vector $1 \in \mathbb{R}^V$ is the eigenvector corresponding to the smallest eigenvalue 0. The generalized eigenvalues of $L_G x = \lambda_i D_G x$ are $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. We will use v_2 to denote smallest non-trivial eigenvector, *i.e.*, the eigenvector corresponding to λ_2; v_3 to denote the next eigenvector; and so on. We will overload notation to use $\lambda_2(A)$ to denote the smallest non-zero generalized eigenvalue of A with respect to D_G. Finally, for a matrix A, let A^+ denote its (uniquely defined) Moore-Penrose pseudoinverse. For two vectors $x, y \in \mathbb{R}^n$, and the degree matrix D_G for a graph G, we define the *degree-weighted inner product* as $x^T D_G y \stackrel{\mathrm{def}}{=} \sum_{i=1}^n x_i y_i d_i$. In particular, if a vector x has unit norm, then $x^T D_G x = 1$. Given a subset of vertices $S \subseteq V$, we denote by 1_S the indicator vector of S in $\mathbb{R}^V$ and by 1 the vector in $\mathbb{R}^V$ having all entries set equal to 1.

3 Optimization Approach to Semi-supervised Eigenvectors

In this section, we provide our main technical results: a motivation and statement of our optimization ansatz; our main algorithm for computing semi-supervised eigenvectors; and an analysis that our algorithm computes solutions of our optimization ansatz.

3.1 Motivation for the Program

Recall the optimization perspective on how one computes the leading nontrivial global eigenvectors of the normalized Laplacian $\mathcal{L}_G$ or, equivalently, of the leading nontrivial generalized eigenvectors of L_G. The first nontrivial eigenvector v_2 is the solution to the problem GlobalSpectral that

GlobalSpectral

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \end{aligned}$$

LocalSpectral

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$$

Generalized LocalSpectral

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G X = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$$

Figure 1: Left: The usual GlobalSpectral partitioning optimization problem; the vector achieving the optimal solution is v_2, the leading nontrivial generalized eigenvector of L_G with respect to D_G. Middle: The LocalSpectral optimization problem, which was originally introduced in [37]; for $\kappa = 0$, this coincides with the usual global spectral objective, while for $\kappa > 0$, this produces solutions that are biased toward the seed vector s. Right: The Generalized LocalSpectral optimization problem we introduce that includes both the locality constraint and a more general orthogonality constraint. Our main algorithm for computing semi-supervised eigenvectors will iteratively compute the solution to Generalized LocalSpectral for a sequence of X matrices. In all three cases, the optimization variable is $x \in \mathbb{R}^n$.

is presented on the left of Figure 1. Equivalently, although GlobalSpectral is a non-convex optimization problem, strong duality holds for it and it's solution may be computed as v_2, the leading nontrivial generalized eigenvector of L_G. (In this case, the value of the objective is λ_2, and global spectral partitioning involves then doing a "sweep cut" over this vector and appealing to Cheeger's inequality.) The next eigenvector v_3 is the solution to GlobalSpectral, augmented with the constraint that $x^T D_G v_2 = 0$; and in general the t^{th} generalized eigenvector of L_G is the solution to GlobalSpectral, augmented with the constraints that $x^T D_G v_i = 0$, for $i \in \{2, \ldots, t-1\}$. Clearly, this set of constraints and the constraint $x^T D_G 1 = 0$ can be written as $x^T D_G X = 0$, where 0 is a $(t-1)$-dimensional all-zeros vector, and where X is an $n \times (t-1)$ orthogonal matrix whose i^{th} column equals v_i (where $v_1 = 1$, the all-ones vector, is the first column of X).

Also presented in Figure 1 is LocalSpectral, which includes a constraint that the solution be well-correlated with an input seed set. This LocalSpectral optimization problem was introduced in [37], where it was shown that the solution to LocalSpectral may be interpreted as a locally-biased version of the second eigenvector of the Laplacian.[1] In particular, although LocalSpectral is not convex, it's solution can be computed efficiently as the solution to a set of linear equations that generalize the popular Personalized PageRank procedure; in addition, by performing a sweep cut and appealing to a variant of Cheeger's inequality, this locally-biased eigenvector can be used to perform locally-biased spectral graph partitioning [37].

[1] In [37], the locality constraint was actually a quadratic constraint, and thus a somewhat involved analysis was required. In retrospect, given the form of the solution, and in light of the discussion below, it is clear that the quadratic part was not "real," and thus we present this simpler form of LocalSpectral here. This should make the connections with our Generalized LocalSpectral objective more immediate.

3.2 Our Main Algorithm

We will formulate the problem of computing semi-supervised vectors in terms of a primitive optimization problem of independent interest. Consider the GENERALIZED LOCALSPECTRAL optimization problem, as shown in Figure 1. For this problem, we are given a graph $G = (V, E)$, with associated Laplacian matrix L_G and diagonal degree matrix D_G; an indicator vector s of a small "seed set" of nodes; a *correlation parameter* $\kappa \in [0, 1]$; and an $n \times \nu$ constraint matrix X that may be assumed to be an orthogonal matrix. We will assume (without loss of generality) that s is properly normalized and orthogonalized so that $s^T D_G s = 1$ and $s^T D_G 1 = 0$. While s can be a general unit vector orthogonal to 1, it may be helpful to think of s as the indicator vector of one or more vertices in V, corresponding to the target region of the graph.

In words, the problem GENERALIZED LOCALSPECTRAL asks us to find a vector $x \in \mathbb{R}^n$ that minimizes the variance $x^T L_G x$ subject to several constraints: that x is unit length; that x is orthogonal to the span of X; and that x is $\sqrt{\kappa}$-well-correlated with the input seed set vector s. In our application of GENERALIZED LOCALSPECTRAL to the computation of semi-supervised eigenvectors, we will iteratively compute the solution to GENERALIZED LOCALSPECTRAL, updating X to contain the already-computed semi-supervised eigenvectors. That is, to compute the first semi-supervised eigenvector, we let $X = 1$, *i.e.*, the n-dimensional all-ones vector, which is the trivial eigenvector L_G, in which case X is an $n \times 1$ matrix; and to compute each subsequent semi-supervised eigenvector, we let the columns of X consist of 1 and the other semi-supervised eigenvectors found in each of the previous iterations.

To show that GENERALIZED LOCALSPECTRAL is efficiently-solvable, note that it is a quadratic program with only one quadratic constraint and one linear equality constraint.[2] In order to remove the equality constraint, which will simplify the problem, let's change variables by defining the $n \times (n - \nu)$ matrix F as

$$\{x : X^T D_G x = 0\} = \{x : x = F\hat{x}\}.$$

That is, F is a span for the null space of X^T; and we will take F to be an orthogonal matrix. In particular, this implies that $F^T F$ is an $(n - \nu) \times (n - \nu)$ Identity and FF^T is an $n \times n$ Projection. Then, with respect to the $\hat{x}$ variable, GENERALIZED LOCALSPECTRAL becomes

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \hat{x}^T F^T L_G F y \\ \text{subject to} \quad & \hat{x}^T F^T D_G F \hat{x} = 1, \\ & \hat{x}^T F^T D_G s \geq \sqrt{\kappa}. \end{aligned} \tag{1}$$

In terms of the variable x, the solution to this optimization problem is of the form

$$\begin{aligned} x^* &= cF \left(F^T (L_G - \gamma D_G) F\right)^+ F^T D_G s \\ &= c \left(FF^T (L_G - \gamma D_G) FF^T\right)^+ D_G s, \end{aligned} \tag{2}$$

for a normalization constant $c \in (0, \infty)$ and for some γ that depends on $\sqrt{\kappa}$. The second line follows from the first since F is an $n \times (n - \nu)$ orthogonal matrix. This so-called "S-proceudre" is described in greater detail in Chapter 5 and Appendix B of [10]. The significance of this is that,

[2] Alternatively, note that it is an example of an constrained eigenvalue problem [20]. We thank the numerous individuals who pointed this out to us subsequent to our dissemination of the original version of this paper.

although it is a non-convex optimization problem, the Generalized LocalSpectral problem can be solved by solving a linear equation, in the form given in Eqn. (2).

Returning to our problem of computing semi-supervised eigenvectors, recall that, in addition to the input for the Generalized LocalSpectral problem, we need to specify a positive integer k that indicates the number of vectors to be computed. In the simplest case, we would assume that we would like the correlation to be "evenly distributed" across all k vectors, in which case we will require that each vector is $\sqrt{\kappa/k}$-well-correlated with the input seed set vector s; but this assumption can easily be relaxed, and thus Algorithm 1 is formulated more generally as taking a k-dimensional vector $\kappa = [\kappa_1, \ldots, \kappa_k]^T$ of correlation coefficients as input.

To compute the first semi-supervised eigenvector, we will let $X = 1$, the all-ones vector, in which case the first nontrivial semi-supervised eigenvector is

$$x_1^* = c\,(L_G - \gamma_1 D_G)^+ D_G s, \tag{3}$$

where γ_1 is chosen to saturate the part of the correlation constraint along the first direction. (Note that the projections FF^T from Eqn. (2) are not present in Eqn. (3) since by design $s^T D_G 1 = 0$.) That is, to find the correct setting of γ_1, it suffices to perform a binary search over the possible values of γ_1 in the interval $(-\mathrm{vol}(G), \lambda_2(G))$ until the correlation constraint is satisfied, that is, until $(s^T D_G x_1)^2$ is sufficiently close to κ_1.

To compute subsequent semi-supervised eigenvectors, *i.e.*, at steps $t = 2, \ldots, k$ if one ultimately wants a total of k semi-supervised eigenvectors, then one lets X be the $n \times t$ matrix of the form

$$X = [1, x_1^*, \ldots, x_{t-1}^*], \tag{4}$$

where $x_1^*, \ldots, x_{t-1}^*$ are successive semi-supervised eigenvectors; and the projection matrix FF^T is of the form

$$FF^T = I - D_G X (X^T D_G D_G X)^{-1} X^T D_G,$$

due to the the degree-weighted inner norm.

Then, by Eqn. (2), the t^{th} semi-supervised eigenvector takes the form

$$x_t^* \;=\; c\left(FF^T(L_G - \gamma_t D_G)FF^T\right)^+ D_G s.$$

In more detail, Algorithm 1 presents pseudo-code for our main algorithm for computing semi-supervised eigenvectors. The algorithm takes as input a graph $G = (V, E)$, a seed set s (which could be a general vector $s \in \mathbb{R}^n$, subject for simplicity to the normalization constraints $s^T D_G 1 = 0$ and $s^T D_G s = 1$, but which is most easily thought of as an indicator vector for the local "seed set" of nodes), a number k of vectors we want to compute, and a vector of locality parameters $(\kappa_1, \ldots, \kappa_k)$, where $\kappa_i \in [0, 1]$ and $\sum_{i=1}^k \kappa_i = 1$ (where, in the simplest case, one could choose $\kappa_i = \kappa/k$, $\forall i$, for some $\kappa \in [0, 1]$). Several things should be noted about our implementation of our main algorithm. First, as we will discuss in more detail below, we compute the projection matrix FF^T only *implicitly*. Second, a naïve approach to Eqn. (2) does not immediately lead to an efficient solution, since $D_G s$ will not be in the span of $(FF^T(L_G - \gamma D_G)FF^T)$, thus leading to a large residual. By changing variables so that $x = FF^T y$, the solution becomes

$$x_t^* \propto FF^T(FF^T(L_G - \gamma_t D_G)FF^T)^+ FF^T D_G s.$$

Since FF^T is a projection matrix, this expression is equivalent to

$$x_t^* \propto \left(FF^T(L_G - \gamma_t D_G)FF^T\right)^+ FF^T D_G s. \tag{5}$$

Algorithm 1 Main algorithm to compute semi-supervised eigenvectors

Require: $L_G, D_G, s, \kappa = [\kappa_1, \ldots, \kappa_k]^T, \epsilon$ such that $s^T D_G 1 = 0$, $s^T D_G s = 1$, $\kappa^T 1 \leq 1$

$X = [1]$
for $t = 1$ to k **do**
  $FF^T \leftarrow I - D_G X (X^T D_G D_G X)^{-1} X^T D_G$
  $\top \leftarrow \lambda_2$ where $FF^T L_G FF^T v_2 = \lambda_2 FF^T D_G FF^T v_2$
  $\perp \leftarrow -\mathrm{vol}(G)$
  repeat
    $\gamma_t \leftarrow (\perp + \top)/2$ (Binary search over γ_t)
    $x_t \leftarrow (FF^T(L_G - \gamma_t D_G)FF^T)^+ FF^T D_G s$
    Normalize x_t such that $x_t^T D_G x_t = 1$
    if $(x_t^T D_G s)^2 > \kappa_t$ **then** $\perp \leftarrow \gamma_t$ **else** $\top \leftarrow \gamma_t$ **end if**
  until $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$ **or** $\|(\perp + \top)/2 - \gamma_t\| \leq \epsilon$
  Augment X with x_t^* by letting $X = [X, x_t^*]$.
end for

Third, regarding the solution x_i, we exploit that $FF^T(L_G - \gamma_i D_G)FF^T$ is an SPSD matrix, and we apply the conjugate gradient method, rather than computing the explicit pseudoinverse. That is, in the implementation we never explicitly represent the dense matrix FF^T, but instead we treat it as an operator and we simply evaluate the result of applying a vector to it on either side. Fourth, we use that λ_2 can never decrease (here we refer to λ_2 as the smallest non-zero eigenvalue of the modified matrix), so we only recalculate the upper bound for the binary search when an iteration saturates without satisfying $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$. Estimating the bound is critical for the semi-supervised eigenvectors to be able to interpolate all the way to the global eigenvectors of the graph, so in Section 3.4 we return to a discussion on efficient strategies for computing the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously computed solutions.

From this discussion, it should be clear that Algorithm 1 solves the semi-supervised eigenvector problem by solving in an iterative manner optimization problems of the form of Generalized LocalSpectral; and that the running time of Algorithm 1 boils down to solving a sequence of linear equations.

3.3 Discussion of Our Main Algorithm

There is a natural "regularization" interpretation underlying our construction of semi-supervised eigenvectors. To see this, recall that the first step of our algorithm can be computed as the solution of a set of linear equations

$$x^* = c\,(L_G - \gamma D_G)^+ D_G s, \tag{6}$$

for some normalization constant c and some γ that can be determined by a binary search over $(-\mathrm{vol}(G), \lambda_2(G))$; and that subsequent steps compute the analogous quantity, subject to additional constraints that the solution be orthogonal to the previously-computed vectors. The quantity $(L_G - \gamma D_G)^+$ can be interpreted as a "regularized" version of the pseudoinverse of L, where $\gamma \in (-\infty, \lambda_2(G))$ serves as the regularization parameter. This interpretation has recently been made precise: [36] show that running a PageRank computation—as well as running other diffusion-based procedures—*exactly* optimizes a regularized version of the GlobalSpectral (or Local-

SPECTRAL, depending on the input seed vector) problem; and [47] provide a precise statistical framework justifying this.

The usual interpretation of PageRank involves "random walkers" who uniformly (or non-uniformly, in the case of Personalized PageRank) "teleport" with a probability $\alpha \in (0,1)$. As described in [37], choosing $\alpha \in (0,1)$ corresponds to choosing $\gamma \in (-\infty, 0)$. By rearranging Eqn. (6) as

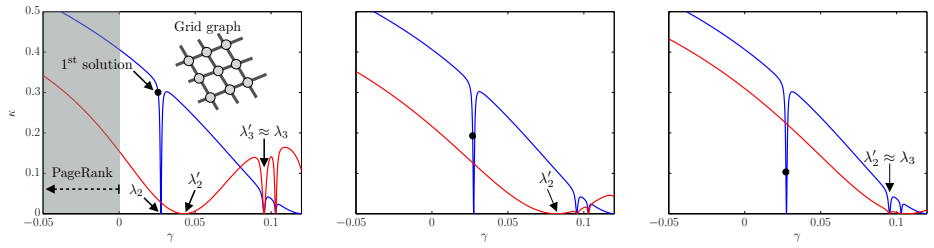
$$\begin{aligned} x^* &= c\left((D_G - A_G) - \gamma D_G\right)^+ D_G s \\ &= \frac{c}{1-\gamma}\left(D_G - \frac{1}{1-\gamma}A_G\right)^+ D_G s \\ &= \frac{c}{1-\gamma} D_G^{-1}\left(I - \frac{1}{1-\gamma}A_G D_G^{-1}\right)^+ D_G s, \end{aligned}$$

we recognize $A_G D_G^{-1}$ as the standard random walk matrix, and it becomes immediate that the solution based on random walkers,

$$x^* = \frac{c}{1-\gamma} D_G^{-1}\left(I + \sum_{i=1}^{\infty}\left(\frac{1}{1-\gamma}D_G^{-1}A_G\right)^i\right) D_G s,$$

is divergent for $\gamma > 0$. Since $\gamma = \lambda_2(G)$ corresponds to no regularization and $\gamma \to -\infty$ corresponds to heavy regularization, viewing this problem in terms of solving a linear equation is formally more powerful than viewing it in terms of random walkers. That is, while all possible values of the regularization parameter—and in particular the (positive) value $\lambda_2(\cdot)$—are achievable algorithmically by solving a linear equation, only values in $(-\infty, 0)$ are achievable by running a PageRank diffusion. In particular, if the optimal value of γ that saturates the κ-dependent locality constraint is negative, then running the PageRank diffusion could find it; otherwise, the "best" one could do will still not saturate the locality constraint, in which case some of the intended correlation is "unused."

An important technical and practical point has to do with the precise manner in which the i^{th} vector is well-correlated with the seed set s. In our formulation, this is captured by a *locality parameter* γ_i that is chosen (via a binary search) to "saturate" the correlation condition, *i.e.*, so that the i^{th} vector is κ/k-well-correlated with the input seed set. As a general rule, successive γ_is need to be chosen that successive vectors are *less* well-localized around the input seed set. (Alternatively, depending on the application, one could choose this parameter so that successive γ_is are equal; but this will involve "sacrificing" some amount of the κ/k correlation, which will lead to the last or last few eigenvectors being very poorly-correlated with the input seed set. These tradeoffs will be described in more detail below.) Informally, if s is a seed set consisting of a small number of nodes that are "nearby" each other, then to maintain a given amount of correlation, we must "view" the graph over larger and larger size scales as we compute more and more semi-supervised eigenvectors. More formally, we need to let the value of the regularization parameter γ at the i^{th} round, we call it γ_i, vary for each $i \in \{1, \ldots, k\}$. That is, γ_i is not pre-specified, but it is chosen via a binary search over the region $(-\mathrm{vol}(G), \lambda_2(\cdot))$, where $\lambda_2(\cdot)$ is the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously-computed vectors (which is in general larger than $\lambda_2(G)$). In this sense, our semi-supervised eigenvectors are both "locally-biased", in a manner that depends on the input seed vector and correlation parameter, and "regularized", in a manner that depends on the local graph structure.



(a) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.3$ (b) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.2$ (c) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.1$

Figure 2: Interplay between the γ parameter and the correlation κ that a semi-supervised eigenvector has with a seed $\boldsymbol{s}$ on a two-dimensional grid. In Figure 2(a)-2(c), we vary the locality parameter for the leading semi-supervised eigenvector, which in each case leads to a value of γ which is marked by the black dot on the blue curve. This allows us to illustrate the influence on the relationship between γ and κ on the next semi-supervised eigenvector. Figure 2(a) also highlights the range ($\gamma < 0$) in which Personalized PageRank can be used for computing solutions to semi-supervised eigenvectors.

To illustrate the previous discussion, Figure 2 considers the two-dimensional grid. In each subfigure, the blue curve shows the correlation with a single seed node as a function of γ for the leading semi-supervised eigenvector, and the black dot illustrates the value of γ for three different values of the locality parameter κ. This relationship between κ and γ is in general non-convex, but it is monotonic for $\gamma \in (-\text{vol}(G), \lambda_2(G))$. The red curve in each subfigure shows the decay for the second semi-supervised eigenvector. Recall that it is perpendicular to the first semi-supervised eigenvector, that the decay is monotonic for $\gamma \in (-\text{vol}(G), \lambda_2'(G))$, and that $\lambda_2 < \lambda_2' \leq \lambda_3$. In Figure 2(a), the first semi-supervised eigenvector is not "too" close to λ_2, and so λ_2' (*i.e.*, the second eigenvalue of the next semi-supervised eigenvector) increases just slightly. In Figure 2(b), we consider a locality parameter that leads to a value of γ that is closer to λ_2, thereby increasing the value of λ_2'. Finally, in Figure 2(c), the locality parameter is such that the leading semi-supervised eigenvector almost coincides with $\boldsymbol{v}_2$; this results in $\lambda_2' \approx \lambda_3$, as required if we were to compute the global eigenvectors.

3.4 Bounding the Binary Search

For the following derivations it is more convenient to consider the normalized graph Laplacian, in which case we define the first solution as

$$y_1 = c\left(\mathcal{L}_G - \gamma_1 I\right)^+ D_G^{1/2} s \tag{7}$$

where $x_1^* = D_G^{-1/2} y_1$. This approach is convenient since the projection operator with null space defined by previous solutions can be expressed as $FF^T = I - YY^T$, assuming that $Y^T Y = 1$. That is, Y is of the form

$$Y = [D_G^{1/2}, y_1^*, \ldots, y_{t-1}^*],$$

where y_i^* are successive solutions to Eqn. (7). In the following the type of projection operator will be implicit from the context, *i.e.*, when working with the combinatorial graph Laplacian $FF^T =$

$I - D_G X(X^T D_G D_G X)^{-1} X^T D_G$, whereas for the normalized graph Laplacian $FF^T = I - YY^T$.

For the normalized graph Laplacian $\mathcal{L}_G$, the eigenvalues of $\mathcal{L}_G v = \lambda v$ equal the eigenvalues of the generalized eigenvalue problem $L_G v = \lambda D_G v$. The binary search employed in Algorithm 1 uses a monotonic relationship between the $\gamma \in (-\mathrm{vol}(G), \lambda_2(\cdot))$ parameter and the correlation with the seed $x^T D_G s$, that can be deduced from the KKT-conditions [37]. Note, that if the upper bound for the binary search $\top = \lambda_2(FF^T \mathcal{L}_G FF^T)$ is not determined with sufficient precision, the search will (if we underestimate $\top$) fail to satisfy the constraint, or (if we overestimate $\top$) fail to converge because the monotonic relationship no longer hold.

By Lemma 1 in Appendix A it follows that $\lambda_2(FF^T \mathcal{L}_G FF^T) = \lambda_2(\mathcal{L}_G + \omega YY^T)$ when $\omega \to \infty$. Since the latter term is a sum of two PSD matrices, the value of the upper bound can only increase as stated by Lemma 2 in Appendix A. This is an important property, because if we do not recalculate $\top$, the previous value is guaranteed to be an underestimate, meaning that the objective will remain convex. Thus, it may be more efficient to first recompute $\top$ when the binary search fails to satisfy $(x^T D_G s)^2 = \kappa$, meaning that $\top$ must be recomputed to increase the search range.

We compute the value for the upper bound of the binary search by transforming the problem in such a way that we can determine the greatest eigenvalue of a new system (fast and robust), and from that, deduce the new value of $\top = \lambda_2(FF^T \mathcal{L}_G FF^T)$. We do so by expanding the expression as

$$\begin{aligned} FF^T \mathcal{L}_G FF^T &= FF^T \left(I - D_G^{-1/2} A_G D_G^{-1/2} \right) FF^T \\ &= FF^T - FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T \\ &= I - \left(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T + YY^T \right). \end{aligned}$$

Since all columns of Y will be eigenvectors of $FF^T \mathcal{L}_G FF^T$ with zero eigenvalue, these will all be eigenvectors of $FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T + YY^T$ with eigenvalue 1. Hence, the largest algebraic eigenvalue $\lambda_{\mathrm{LA}}(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T)$ can be used to compute the upper bound for the binary search as

$$\top = \lambda_2(FF^T \mathcal{L}_G FF^T) = 1 - \lambda_{\mathrm{LA}}(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T). \tag{8}$$

The reason for not considering the largest magnitude eigenvalue, is that A_G may be indefinite. Finally, with respect to our implementation we emphasize that FF^T is used as a projection operator, and not represented explicitly.

4 Extension of Our Main Algorithm and Implementation Details

In this section, we present two variants of our main algorithm that are more well-suited for very large-scale applications; the first uses a column-based low-rank approximation, and the second uses random walk ideas. In Section 4.1, we describe how to use the Nyström method, which constructs a low-rank approximation to the kernel matrix by sampling columns, to construct a general solution for semi-supervised eigenvectors, where the low-rankness is exploited for very efficient computation. Then, in Section 4.2, we describe a "Push-peeling heuristic," based on the efficient Push algorithm by [1]. The basic idea is that if, rather than iteratively computing locally-biased semi-supervised eigenvectors using the procedure described in Algorithm 1, we instead compute solutions to LocalSpectral and then construct the semi-supervised eigenvectors by

"projecting away" pieces of these solutions, then we can take advantage of local random walks that have improved algorithmic properties.

4.1 A Nyström-based Low-rank Approach

Here we describe the use of the recently-popular Nyström method to speed up the computation of semi-supervised eigenvectors. We do so by considering how a low-rank decomposition can be exploited to yield solutions to the Generalized LocalSpectral objective in Figure 1, where the running time largely depends on a matrix-vector product. These methods are most appropriate when the kernel matrix is reasonably well-approximated by a low-rank matrix [15, 22, 64].

Given some low-rank approximation $\mathcal{L}_G \approx I - V\Lambda V^T$, we apply the Woodbury matrix identity, and we derive an explicit solution for the leading semi-supervised eigenvector

$$\begin{aligned} y_1 &\approx c\left((1-\gamma)I - V\Lambda V^T\right)^+ D_G^{1/2}s \\ &\approx c\left(\frac{1}{1-\gamma}I + \frac{1}{(1-\gamma)^2}V\left(\Lambda^{-1} - \frac{1}{1-\gamma}I\right)^{-1}V^T\right)D_G^{1/2}s \\ &\approx \frac{c}{1-\gamma}\left(I + V\Sigma V^T\right)D_G^{1/2}s, \end{aligned}$$

where $\Sigma_{ii} = \frac{1}{\frac{1-\gamma}{\lambda_i}-1}$. In order to compute efficiently the subsequent semi-supervised eigenvectors we must accommodate for the projection operator $FF^T = I - YY^T$, while yet exploiting the explicit closed-form inverse $(\mathcal{L}_G - \gamma I)^+ \approx \frac{1}{1-\gamma}\left(I + V\Sigma V^T\right)$. However, the projection operator complicates the expression, since the previous solution can be spanned by multiple global eigenvectors, so leveraging from the low-rank decomposition is more difficult for the inverse $(FF^T(\mathcal{L}_G - \gamma I)FF^T)^+$.

Conveniently, we can decouple the projection operator by treating the orthogonality constraint using a Lagrangian approach, such that the solution can be expressed as

$$y_t = c\left(\mathcal{L}_G - \gamma I + \omega YY^T\right)^+ D_G^{1/2}s,$$

where $\omega \geq 0$ denotes the associated Lagrange multiplier, and where the sign is deduced from the KKT conditions. Applying the Woodbury matrix identity is now straightforward

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega Y^T {P_\gamma}^+ Y\right)^+ Y^T {P_\gamma}^+,$$

where for notational convenience we have introduced $P_\gamma = \mathcal{L}_G - \gamma I$. By decomposing $Y^T {P_\gamma}^+ Y$ with an eigendecomposition USU^T the equation simplifies as follows

$$\begin{aligned} \left(P_\gamma + \omega YY^T\right)^+ &= {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega USU^T\right)^+ Y^T {P_\gamma}^+ \\ &= {P_\gamma}^+ - {P_\gamma}^+ YU\Omega U^T Y^T {P_\gamma}^+, \end{aligned}$$

where $\Omega_{ii} = \frac{1}{\frac{1}{\omega}+S_{ii}}$. Note how this result gives a well defined way of controlling the amount of "orthogonality", and by Lemma 1 in Appendix A, we get exact orthogonality in the limit of $\omega \to \infty$, in which case the expression simplifies to

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - {P_\gamma}^+ Y(Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+.$$

Using the explicit expression for ${P_\gamma}^+$, the solution now only involves matrix-vector products and the inverse of a small matrix

$$y_t = c\left({P_\gamma}^+ - {P_\gamma}^+ Y (Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+\right) D_G^{1/2} s. \tag{9}$$

To conclude this section, let us also consider how we can optimize the efficiency of the calculation of $\lambda_2(FF^T \mathcal{L}_G FF^T)$ used for bounding the binary search in Algorithm 1. According to Eqn. (8) the bound can be calculated efficiently as $\top = 1 - \lambda_{\mathrm{LA}}(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T)$. However, by substituting with $D_G^{-1/2} A_G D_G^{-1/2} \approx V \Lambda V^T$, we can exploit low-rankness since

$$\top = 1 - \lambda_{\mathrm{LA}}(FF^T V \Lambda V^T FF^T) = 1 - \lambda_{\mathrm{LA}}(\Lambda^{1/2} V^T FF^T V \Lambda^{1/2}),$$

where the latter is a much smaller system.

4.2 A Push-peeling Heuristic

Here we present a variant of our main algorithm that exploits the connections between diffusion-based procedures and eigenvectors, allowing semi-supervised eigenvectors to be efficiently computed for large networks. This is most well-known for the leading nontrivial eigenvectors of the graph Laplacian [12]; but recent work has exploited these connections in the context of performing locally-biased spectral graph partitioning [55, 1, 37]. In particular, we can compute the locally-biased vector using the first step of Algorithm 1, or alternatively we can compute it using a locally-biased random walk of the form used in [55, 1]. Here we present a heuristic that works by peeling off components from a solution to the PageRank problem, and by exploiting the regularization interpretation of γ, we can from these components obtain the subsequent semi-supervised eigenvectors.

Specifically, we focus on the Push algorithm by [1]. This algorithm approximates the solution to PageRank very efficiently, by exploiting the local modifications that occur when the seed is highly concentrated. This makes our algorithm very scalable and applicable for large-scale data, since only the local neighborhood near the seed set will be touched by the algorithm. In comparison, by solving the linear system of equations we explicitly touch all nodes in the graph, even though most spectral rankings will be below the computational precision [8].

We adapt a similar notation as in [1] and start by defining the usual PageRank vector $\mathrm{pr}(\alpha, s_{\mathrm{pr}})$ as the unique solution of the linear system

$$\mathrm{pr}(\alpha, s_{\mathrm{pr}}) = \alpha s_{\mathrm{pr}} + (1-\alpha) A_G D_G^{-1} \mathrm{pr}(\alpha, s_{\mathrm{pr}}), \tag{10}$$

where α is the teleportation parameter, and s_{pr} is the sparse starting vector. For comparison, the push algorithm by [1] computes an approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$ for a slightly different system

$$\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}) = \alpha' s_{\mathrm{pr}} + (1-\alpha') W \mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}),$$

where $W = \frac{1}{2}(I + A_G D_G^{-1})$ and not the usual random walk matrix $A D_G^{-1}$ as used in Eqn. (10). However, these equations are only superficially different, and equivalent up to a change of the respective teleportation parameter. Thus, it is straightforward to verify that these teleportation parameters and the γ parameter of Eqn. (6) are related as

$$\alpha = \frac{2\alpha'}{1+\alpha'} \Leftrightarrow \alpha' = \frac{\alpha}{2-\alpha} \Leftrightarrow \alpha' = \frac{\gamma}{\gamma - 2},$$

and that the leading semi-supervised eigenvector for $\gamma \in (-\infty, 0)$ can be approximated as

$$x_1^* \approx \frac{c}{-\gamma} D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma}{\gamma - 2}, D_G s \right).$$

To generalize subsequent semi-supervised eigenvectors to this diffusion based framework, we need to accommodate for the projection operator such that subsequent solutions can be expressed in terms of graph diffusions. By requiring distinct values of γ for all semi-supervised eigenvectors, we may use the solution for the leading semi-supervised eigenvector and then systematically "peel off" components, thereby obtaining the solution of one of the consecutive semi-supervised eigenvectors. By Lemma 5, in Appendix A the general solution in Eqn. (5) can be approximated by

$$x_t^* \approx c \left(I - X X^T D_G \right) \left(L_G - \gamma_t D_G \right)^+ D_G s, \tag{11}$$

under the assumption that all γ_k for $1 < k \leq t$ are sufficiently apart. If we think about γ_k as being distinct eigenvalues of the generalized eigenvalue problem $L_G x_k = \gamma_k D_G x_k$, then it is clear that Eqn. (11), correctly computes the sequence of generalized eigenvectors. This is explained by the fact that $(L_G - \gamma_t D_G)^+ D_G s$ can be interpreted as the first step of the Rayleigh quotient iteration, where γ_t is the estimate of the eigenvalue, and $D_G s$ is the estimate of the eigenvector. Given that the estimate of the eigenvalue is right, this algorithm will in the initial step compute the corresponding eigenvector, and the operator $\left(I - X X^T D_G \right)$ will be superfluous, as the global eigenvectors are already orthogonal in the degree-weighted norm. To quantify the failure modes of the approximation, let us consider what happens when γ_2 starts to approach γ_1. What constitutes the second solution for a particular value of γ_2 is the perpendicular component with respect to the projection onto the solution given by γ_1. As γ_2 approaches γ_1, this perpendicular part diminishes and the solution becomes ill-posed. Fortunately, we can easily detect such issues during the binary search in Algorithm 1, and in general the approximation has turned out to work very well in practice as our experimental results in Section 5 show.

In terms of the approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$, the general approximate solution takes the following form

$$x_t^* \approx c \left(I - X X^T D_G \right) D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma_t}{\gamma_t - 2}, D_G s \right). \tag{12}$$

As already stated in Section 3.3, the impact of using a diffusion based procedure is that we cannot interpolate all the way to the global eigenvectors, and that the main challenge is that the solutions do not become too localized. The ϵ parameter of the Push algorithm controls the threshold for propagating mass away from the seed set and into the adjacent nodes in the graph. If the threshold is too high, the solution will be very localized and make it difficult to find more than a few semi-supervised eigenvectors, as characterized by Lemma 3 in Appendix A, because the leading ones will then span the entire space of the seed set. As the choice of ϵ is important for the applicability of our algorithm, we will in Section 5 investigate the influence of this parameter on large data graphs.

To conclude this section, we consider an important implementation detail that have been omitted so far. In the work of [37] the seed vector was defined to be perpendicular to the all-ones vector, and for the sake of consistency we have chosen to define it in the same way. The impact of projecting the seed set to a space that is perpendicular to the all-ones vector is that the resulting seed vector is no longer sparse, making the use of the Push algorithm in Eqn. (12) inefficient. The seed vector can, however, without loss of generality, be defined as $s \propto D_G^{-1/2} \left(I - v_0 v_0^T \right) s_0$ where s_0 is

the sparse seed, and $v_0 \propto \mathrm{diag}\left(D_G^{1/2}\right)$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-ones vector of the combinatorial graph Laplacian). If we substitute with this expression for the seed in Eqn. (12), it follows by plain algebra (see Appendix B) that

$$x_t^* \approx c\left(I - XX^T D_G\right)\left(D_G^{-1}\mathrm{pr}_\epsilon\left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2}s_0\right) - D_G^{-1/2}v_0 v_0^T s_0\right). \tag{13}$$

Now the Push algorithm is only defined on the sparse seed set making the the expression very scalable. Finally, the Push algorithm maintains a queue of high residual nodes that are yet to be processed. The order in which nodes are processed influences the overall running time, and in [8] preliminary experiments showed that a FIFO queue resulted in the best performance for large values of γ, as compared to a priority queue that scales logarithmically. For this reason we have chosen to use a FIFO queue in our implementation.

5 Empirical Results

In this section, we provide a detailed empirical evaluation of the method of semi-supervised eigenvectors and how it can be used for locally-biased machine learning. Our goal is two-fold: first, to illustrate how the "knobs" of the method work; and second, to illustrate the usefulness of the method in real applications. To do this, we consider several classes of data.

- **Toy data.** In Section 5.1, we consider one-dimensional examples of the popular "small world" model [62]. This is a parameterized family of models that interpolates between low-dimensional grids and random graphs; and, as such, it allows us to illustrate the behavior of the method and its various parameters in a controlled setting.
- **Congressional voting data.** In Section 5.2, we consider roll call voting data from the United States Congress that are based on [49]. This is an example of realistic data set that has relatively-simple global structure but nontrivial local structure that varies with time [14]; and thus it allows us to illustrate the method in a realistic but relatively-clean setting.
- **Handwritten image data.** In Section 5.3, we consider data from the MNIST digit data set [34]. These data have been widely-studied in machine learning and related areas and they have substantial "local heterogeneity." Thus, these data allow us to illustrate how the method may be used to perform locally-biased versions of common machine learning tasks such as smoothing, clustering, and kernel construction.
- **Functional brain imaging data.** In Section 5.4, we consider functional magnetic resonance imaging (fMRI) data. Single subject fMRI data is characterized by high dimensionality and relatively few samples, in contrast to the MNIST data that consist of many samples and a relatively low dimensionality. We demonstrate how our semi-supervised eigenvectors can be applied to construct a data-driven spatially-biased basis by incorporating *a priori* knowledge from a functional brain atlas [17].
- **Large-scale network data.** In Section 5.5, we consider large-scale network data, and demonstrate significant performance improvements of the push-peeling heuristic compared to solving the same equations using a conjugate gradient solver. These improvements are demonstrated on datasets from the DIMACS implementation challenge, as well as on large web-crawls with more then 3 billion non-zeros in the adjacency matrix [44, 45, 46].

5.1 Small-world Data

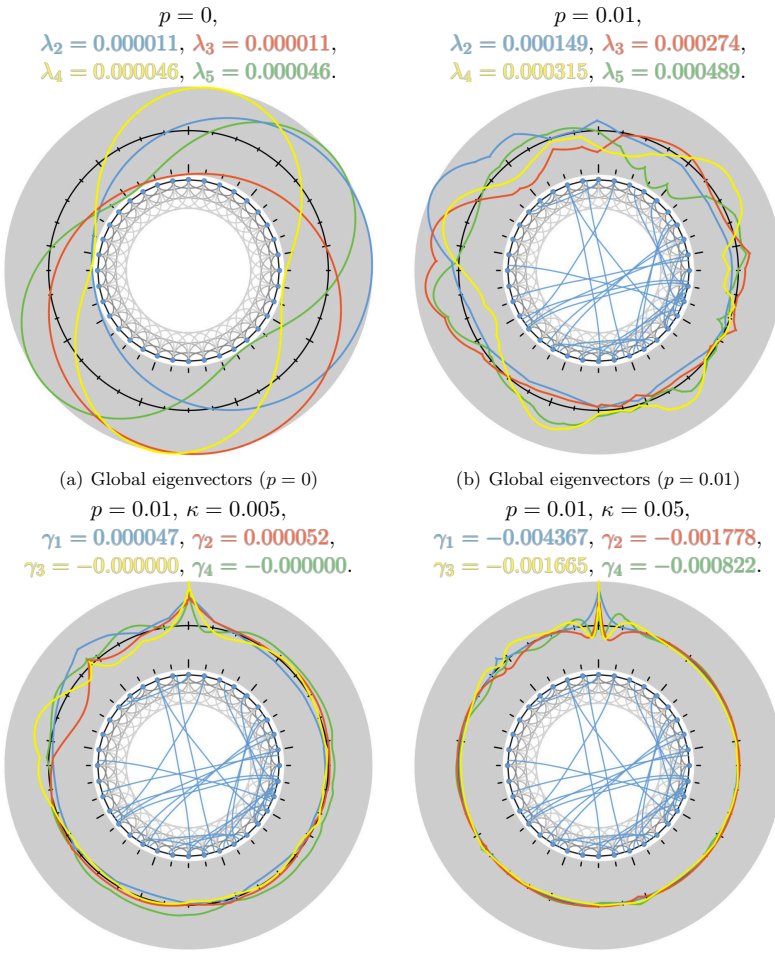
The first data sets we consider are networks constructed from the so-called small-world model. This model can be used to demonstrate how semi-supervised eigenvectors focus on specific target regions of a large data graph to capture slowest modes of local variation; and it can also be used to illustrate how the "knobs" of the method work, *e.g.*, how κ and γ interplay, in a practical setting. In Figure 3, we plot the usual global eigenvectors, as well as locally-biased semi-supervised eigenvectors, around illustrations of non-rewired and rewired realizations of the small-world graph, *i.e.*, for different values of the rewiring probability p and for different values of the locality parameter κ.

To start, in Figure 3(a) that we show a graph with no randomly-rewired edges ($p = 0$) and a parameter κ such that the global eigenvectors are obtained. This yields a symmetric graph with eigenvectors corresponding to orthogonal sinusoids, *i.e.*, the first two capture the slowest mode of variation and correspond to a sine and cosine with equal random phase-shift (up to a rotational ambiguity). In Figure 3(b), random edges have been added with probability $p = 0.01$ and the parameter κ is still chosen such that the global eigenvectors—now of the rewired graph—are obtained. Note the many small kinks in the eigenvectors at the location of the randomly added edges. Note also the slow mode of variation in the interval on the top left; a normalized-cut based on the leading global eigenvector would extract this region, since the remainder of the ring is more well-connected due to the random rewiring.

In Figure 3(c), we see the same graph realization as in Figure 3(b), except that the semi-supervised eigenvectors have a seed node at the top of the circle, *i.e.*, at "12 o-clock," and the locality parameter $\kappa_t = 0.005$, which corresponds to moderately well-localized eigenvectors. As with the global eigenvectors, the locally-biased semi-supervised eigenvectors are of successively-increasing (but still localized) variation. Note also that the neighborhood around "11 o-clock" contains more mass, *e.g.*, when compared with the same parts of the circle in Figure 3(b) or with other parts of the circle in Figure 3(c), even though it is not very near the seed node in the original graph geometry. The reason for this is that this region is well-connected with the seed via a randomly added edge, and thus it is close in the modified graph topology. Above this visualization, we also show the value of γ_t that saturates κ_t, *i.e.*, γ_t is the Lagrange multiplier that defines the effective locality κ_t. Not shown is that if we kept reducing κ_t, then γ_t would tend towards λ_{t+1}, and the respective semi-supervised eigenvectors would tend towards the global eigenvectors that are illustrated in Figure 3(b). Finally, in Figure 3(d), the desired locality is increased to $\kappa = 0.05$ (which has the effect of decreasing the value of γ_t), making the semi-supervised eigenvectors more localized in the neighborhood of the seed. It should be clear that, in addition to being determined by the locality parameter, we can think of γ as a regularizer biasing the global eigenvectors towards the region near the seed set. That is, variation in eigenvectors that are near the initial seed (in the modified graph topology) are most important, while variation that is far away from the initial seed matters much less.

5.2 Congressional Voting Data

The next data set we consider is a network constructed from a time series of roll call voting patterns from the United States Congress that are based on [49]. This is a particularly well-structured social network for which there is a great deal of meta-information, and it has been studied recently with graph-based methods [40, 63, 14]. Thus, it permits a good illustration of the method of semi-supervised eigenvectors in a real application [48]. This data set is known to have nontrivial time-



(a) Global eigenvectors ($p = 0$)

(b) Global eigenvectors ($p = 0.01$)

(c) Semi-supervised eigenvectors

(d) Semi-supervised eigenvectors

Figure 3: Illustration of small-world graphs with rewiring probability of $p = 0$ or $p = 0.01$ and with different values of the κ parameter. For each subfigure, the data consist of 3600 nodes, each connected to it's 8 nearest-neighbors. In the center of each subfigure, we show the nodes (blue) and edges (black and light gray are the local edges, and blue are the randomly-rewired edges). We wrap around the plots (black x-axis and gray background), visualizing the 4 smallest semi-supervised eigenvectors. Eigenvectors are color coded as **blue**, **red**, **yellow**, and **green**, starting with the one having the smallest eigenvalue.

varying structure at different time steps, and we will illustrate how the method of semi-supervised eigenvectors can perform locally-biased classification with a traditional kernel-based algorithm.

In more detail, we evaluate our method by considering the known Congress data-set containing the roll call voting patterns in the U.S Senate across time. We considered Senates in the 70^{th} Congress through the 110^{th} Congress, thus covering the years 1927 to 2008. During this time, the U.S went from 48 to 50 states, hence the number of senators in each of these 41 Congresses was roughly the same. We constructed an $N \times N$ adjacency matrix, with $N = 4196$ (41 Congresses each with ≈ 100 Senators) where $A_{ij} \in [0, 1]$ represents the extent of voting agreement between legislators i and j, and where identical senators in adjacent Congresses are connected with an inter-Congress connection strength. We then considered the Laplacian matrix of this graph, constructed in the usual way [14].

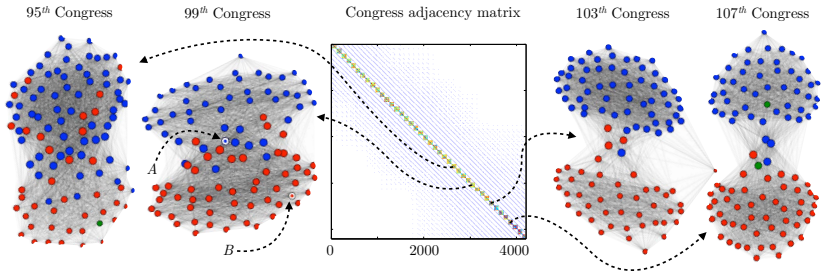


Figure 4: Shows the Congress adjacency matrix, along with four of the individual Congresses. Nodes are scaled according to their degree, blue nodes correspond to Democrats, red to Republicans, and green to Independents.

Figure 4 visualizes the adjacency matrix, along with four of the individual Congresses, color coded by party. This illustrates that these data should be viewed—informally—as a structure (depending on the specific voting patterns of each Congress) evolving along a one-dimensional temporal axis, confirming the results of [14]. Note that the latter two Congresses are significantly better described by a simple two-clustering than the former two Congresses, and an examination of the clustering properties of each of the 40 Congresses reveals significant variation in the local structure of individual Congresses, in a manner broadly consistent with [48] and [49]. In particular, the more recent Congresses are significantly more polarized.

The first vertical column of Figure 5 illustrates the first three global eigenvectors of the full data set, illustrating fluctuations that are sinusoidal and consistent with the one-dimensional temporal scaffolding. Also shown in the first column are the values of that eigenfunction for the members of the 99^{th} Congress, illustrating that there is *not* a good separation based on party affiliations. The next three vertical columns of Figure 5 illustrate various localized eigenvectors computed by starting with a seed node in the 99^{th} Congress. For the second column, we visualize the semi-supervised eigenvectors for a very low correlation ($\kappa = 0.001$), which corresponds to only a weak localization—in this case one sees eigenvectors that look very similar to the global eigenvectors, and the elements of the eigenvector on that Congress do not reveal partitions based on the party cuts.

The third and fourth column of Figure 5 illustrate the semi-supervised eigenvectors for a much higher correlation ($\kappa = 0.1$), meaning a much stronger amount of locality. In particular, the third

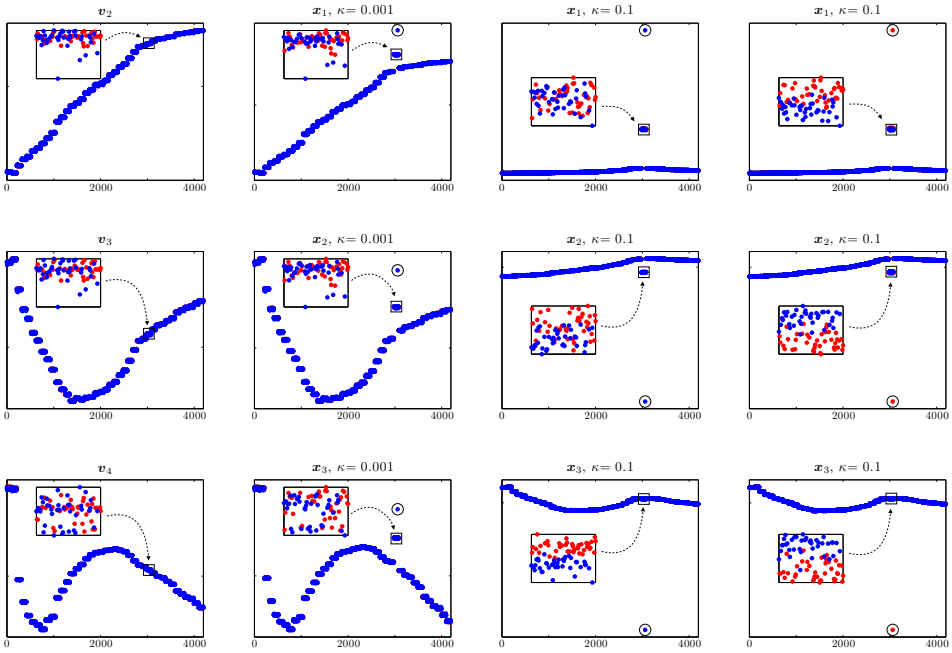


Figure 5: First column: The leading three nontrivial global eigenvectors. Second column: The leading three semi-supervised eigenvectors seeded (circled node) in an articulation point between the two parties in the 99^{th} Congress (see Figure 4), for correlation $\kappa = 0.001$. Third column: Same seed as previous column, but for a correlation of $\kappa = 0.1$. Notice the localization on the third semi-supervised eigenvector. Fourth column: Same correlation as the previous column, but for another seed node well within the cluster of Republicans. Notice the localization on all three semi-supervised eigenvectors.

column starts with the seed node marked A in Figure 4, which is at the articulation point between the two parties, while the fourth column starts with the seed node marked B, which is located well within the cluster of Republicans. In both cases the eigenvectors are much more strongly localized on the 99^{th} Congress near the seed node, and in both cases one observes the partition into two parties based on the elements of the localized eigenvectors. Note, however, that when the initial seed is at the articulation point between two parties then the situation is much noisier: in this case, this "partitionability" is seen only on the third semi-supervised eigenvector, while when the initial seed is well within one party then this is seen on all three eigenvectors. Intuitively, when the seed set is strongly within a good cluster, then that cluster tends to be found with semi-supervised eigenvectors (and we will observe this again below). This is consistent with the diffusion interpretation of eigenvectors. This is also consistent with [14], who observed that the properties of eigenvector localization depended on the local structure of the data around the seed node, as well as the larger scale structure around that local cluster.

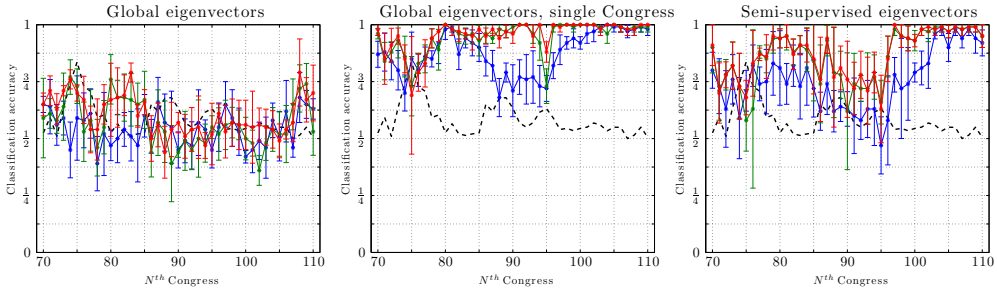


Figure 6: Classification accuracy measured in individual Congresses. For each Congress we perform 5-fold cross validation based on ≈ 80 samples and leave out the remaining 20 samples to estimate an unbiased test error. Error bars are obtained by resampling and they correspond to 1 standard deviation. For each approach we consider features based on the 1^{st} (blue), 2^{nd} (green), and 3^{rd} (red) smallest eigenvector(s), excluding the all-one vector. We also plot the probability of the most probable class as a baseline measure (black) as some Congresses are very imbalanced.

To illustrate how these structural properties manifest themselves in a more traditional machine learning task, we also consider the classification task of discriminating between Democrats and Republicans in single Congresses, *i.e.*, we measure to what extent we can extract local discriminative features. To do so, we apply L_2-regularized L_2-loss support vector classification with a linear kernel, where features are extracted using the global eigenvectors of the entire data set, global eigenvectors from a single Congress (best case measure), and our semi-supervised eigenvectors. Figure 6 illustrates the classification accuracy for 1, 2, and 3 eigenvectors. As reported by [14], locations that exhibit discriminative information are best found on low-order eigenvectors of this data, explaining why the classifier based global eigenvectors performs poorly. In the classifier based on global eigenvectors in the single Congress we exploit *a priori* knowledge to extract the relevant data, that in a usual situation would be impossible. Hence, this is simply to define a baseline point of reference for the best case classification accuracy. The classifier based on semi-supervised eigenvectors is seeded using a few training samples and performs in-between the two other approaches. Compared to our point of reference, Congresses in the range 88 to 96 do worse with the semi-supervised eigenvectors; whereas for Congresses after 100 the semi-supervised approach almost performs on par, even for a single single eigenvector. This is consistent with the visualization in Figure 4 illustrating that earlier Congresses are less cleanly separable, as well as with empirical evidence indicating heterogeneity due to Southern Democrats in earlier Congresses and the recent increase in party polarization in more recent Congresses, as described in [48] and [49].

5.3 MNIST Digit Data

The next data set we consider is the well-studied MNIST data set containing $60,000$ training digits and $10,000$ test digits ranging from 0 to 9; and, with these data, we demonstrate the use of semi-supervised eigenvectors as a feature extraction preprocessing step in a traditional machine learning setting. We construct the full $70,000 \times 70,000$ k-NN graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$, where σ_i^2 is the Euclidian distance of the i^{th} node

to it's nearest neighbor; and from this we define the graph Laplacian in the usual way. We then evaluate the semi-supervised eigenvectors in a transductive learning setting by disregarding the majority of labels in the entire training data. We use a few samples from each class to seed our semi-supervised eigenvectors as well as a few others to train a downstream classification algorithm. For this evaluation, we use the Spectral Graph Transducer (SGT) of [29]; and we choose to use it for two main reasons. First, the transductive classifier is inherently designed to work on a subset of global eigenvectors of the graph Laplacian, making it ideal for validating that the localized basis constructed by the semi-supervised eigenvectors can be more informative when we are solely interested in the "local heterogeneity" near a seed set. Second, using the SGT based on global eigenvectors is a good point of comparison, because we are only interested in the effect of our subspace representation. (If we used one type of classifier in the local setting, and another in the global, the classification accuracy that we measure would obviously be confounded.) As in [29], we normalize the spectrum of both global and semi-supervised eigenvectors by replacing the eigenvalues with some monotonically increasing function. We use $\lambda_i = \frac{i^2}{k^2}$, *i.e.*, focusing on ranking among smallest cuts; see [11]. Furthermore, we fix the regularization parameter of the SGT to $c = 3200$, and for simplicity we fix $\gamma = 0$ for all semi-supervised eigenvectors, implicitly defining the effective $\kappa = [\kappa_1, \ldots, \kappa_k]^T$. Clearly, other correlation distributions κ and other values of γ parameter may yield subspaces with even better discriminative properties (which is an issue to which we will return in Section 5.3.2 in greater detail).

	#Semi-supervised eigenvectors for SGT						#Global eigenvectors for SGT					
Labeled points	1	2	4	6	8	10	1	5	10	15	20	25
1 : 1	0.39	0.39	0.38	0.38	0.38	0.36	0.50	0.48	0.36	0.27	0.27	0.19
1 : 10	0.30	0.31	0.25	0.23	0.19	0.15	0.49	0.36	0.09	0.08	0.06	0.06
5 : 50	0.12	0.15	0.09	0.08	0.07	0.06	0.49	0.09	0.08	0.07	0.05	0.04
10 : 100	0.09	0.10	0.07	0.06	0.05	0.05	0.49	0.08	0.07	0.06	0.04	0.04
50 : 500	0.03	0.03	0.03	0.03	0.03	0.03	0.49	0.10	0.07	0.06	0.04	0.04

Table 1: Classification error for discriminating between 4s and 9s for the SGT based on, respectively, semi-supervised eigenvectors and global eigenvectors. The first column from the left encodes the configuration, *e.g.*, 1:10 interprets as 1 seed and 10 training samples from each class (total of 22 samples—for the global approach these are all used for training). When the seed is well-determined and the number of training samples moderate (50:500), then a single semi-supervised eigenvector is sufficient; whereas for less data, we benefit from using multiple semi-supervised eigenvectors. All experiments have been repeated 10 times.

5.3.1 Discriminating between pairs of digits

Here, we consider the task of discriminating between two digits; and, in order to address a particularly challenging task, we work with $4s$ and $9s$. (This is particularly challenging since these two classes tend to overlap more than other combinations since, *e.g.*, a closed 4 can resemble a 9 more than an open 4.) Hence, we expect that the class separation axis will not be evident in the leading global eigenvector, but instead it will be "buried" further down the spectrum; and we hope to find a "locally-biased class separation axis" with locally-biased semi-supervised eigenvectors. Thus, this example will illustrate how semi-supervised eigenvectors can represent relevant heterogeneities in a local subspace of low dimensionality. See Table 1, which summarizes our classification results based on, respectively, semi-supervised eigenvectors and global eigenvectors, when we use the SGT. See also Figure 7 and Figure 8, which illustrate two realizations for the 1:10 configuration. In

these two figures, the training samples are fixed; and, to demonstrate the influence of the seed, we have varied the seed nodes. In particular, in Figure 7 the seed nodes s_+ and s_- are located well-within the respective classes; while in Figure 8, they are located much closer to the boudary between the two classes. As intuitively expected, when the seed nodes fall well within the classes to be differentiated, the classification is much better than when the seed nodes are located closer to the boundary between the two classes. See the caption in these figures for further details.

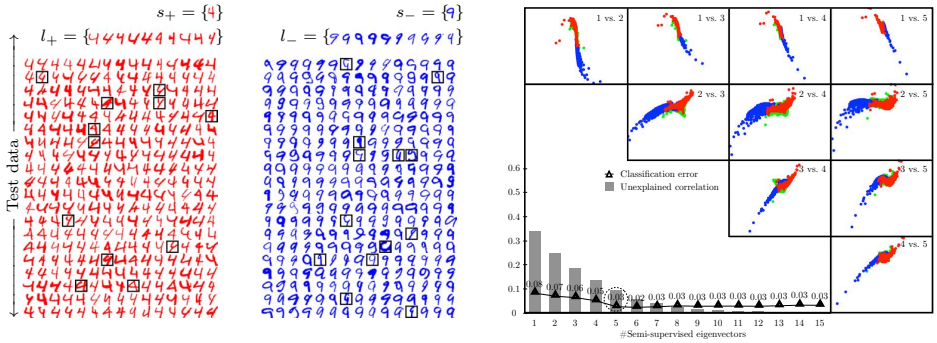


Figure 7: Discrimination between 4s and 9s. Left: Shows a subset of the classification results for the SGT based on 5 semi-supervised eigenvectors seeded in s_+ and s_-, and trained using samples l_+ and l_-. Misclassifications are marked with black frames. Right: Visualizes all test data spanned by the first 5 semi-supervised eigenvectors, by plotting each component as a function of the others. Red (blue) points correspond to 4 (9), whereas green points correspond to remaining digits. As the seed nodes are good representatives, we note that the eigenvectors provide a good class separation. We also plot the error as a function of local dimensionality, as well as the unexplained correlation, *i.e.*, initial components explain the majority of the correlation with the seed (effect of $\gamma = 0$). The particular realization based on the leading 5 semi-supervised eigenvectors yields an error of ≈ 0.03 (dashed circle).

5.3.2 Effect of choosing the κ correlation/locality parameter

Here, we discuss the effect of the choice of the correlation/locality parameter κ at different steps of Algorithm 1, *e.g.*, how $\{\kappa_t\}_{t=1}^k$ should be distributed among the k components. For example, will the downstream classifier benefit the most from a uniform distribution or will there exist some other nonuniform distribution that is better? Although this will be highly problem specific, one might hope that in realistic applications the classification performance is not too sensitive to the actual choice of distribution. To investigate the effect in our example of discriminating between 4s and 9s, we consider 3 semi-supervised eigenvectors for various κ distributions. Our results are summarized in Figure 9.

Figures 9(a), 9(b), and 9(c) show, for the global eigenvectors and for semi-supervised eigenvectors, where the κ vector has been chosed to be very nonuniform and very uniform, the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and Figure 9(d) shows the test error as the

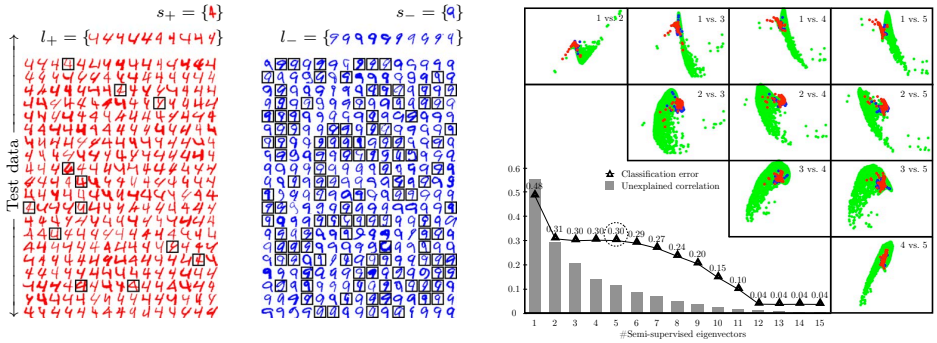
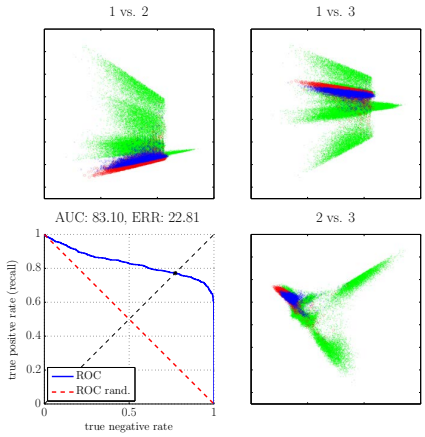


Figure 8: Discrimination between 4s and 9s. See the general description in Figure 7. Here we illustrate an instance where the s_+ shares many similarities with s_-, *i.e.*, s_+ is on the boundary of the two classes. This particular realization achieves a classification error of ≈ 0.30 (dashed circle). In this constellation we first discover localization on low order semi-supervised eigenvectors (≈ 12 eigenvectors), which is comparable to the error based on global eigenvectors (see Table 1), *i.e.*, further down the spectrum we recover from the bad seed and pickup the relevant mode of variation.

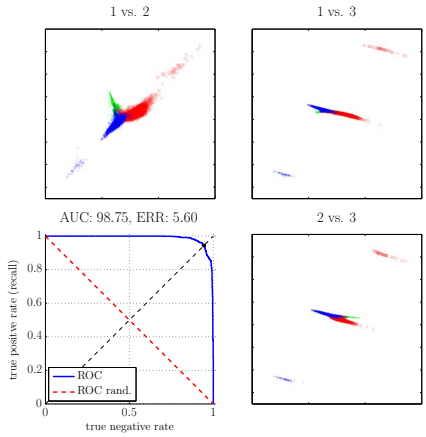
κ vector is varied over the unit simplex. In more detail, red (respectively, blue) corresponds to 4s (respectively, 9s), and green points are the remaining digits; and, for Figures 9(b) and 9(c), the semi-supervised eigenvectors are seeded using 50 samples from each target class (4s vs. 9s) and having a non-uniform distribution of κ, as specified. As seen from the visualization of the semi-supervised eigenvectors in Figures 9(b) and 9(c), the classes are much better separated than by using the global eigenvectors, which are shown in Figure 9(a). For example, this is supported by the Area Under the Curve (AUC) and Error Rate (ERR), being the point on the Receiver Operating Characteristic (ROC) curve that corresponds to having an equal probability of miss-classifying a positive or negative sample, which is a fair estimate as the classes in the MNIST data set is fairly balanced. For Figure 9(c), where we use a uniform distribution of κ, the classifier performs slightly better than in Figure 9(b), which uses the non-uniform κ distribution (but both semi-supervised approaches are significantly better than the using the global eigenvectors). For Figure 9(d), we see the test error on the simplex defined by κ. To obtain this plot we sampled 500 different κ distributions according to a uniform Dirichlet distribution. With the exception of one extreme very nonuiform corner, the classification accuracy is not too sensitive to the choice of κ distribution. Thus, if we think of the semi-supervised eigenvectors as a locally-regularized version of the global eigenvectors, the desired discriminative properties are not too sensitive to the details of the locally-biased regularization.

5.3.3 Effect of approximately computing semi-supervised eigenvectors

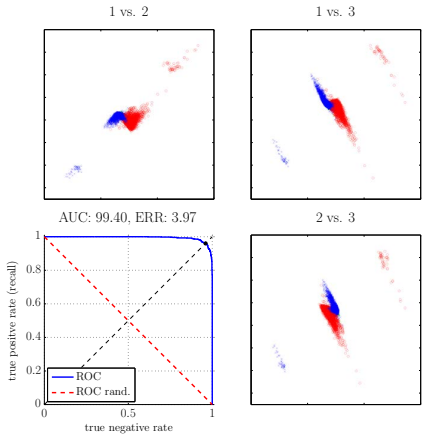
Here, we discuss of the push-peeling procedure from Section 4 that is designed to compute efficient approximations to the semi-supervised eigenvectors by using local random walks to compute an approximation to personalized PageRank vectors. Consider Figure 10, which shows results for two values of the ϵ parameter (*i.e.*, the parameter in the push algorithm that implicitly determines



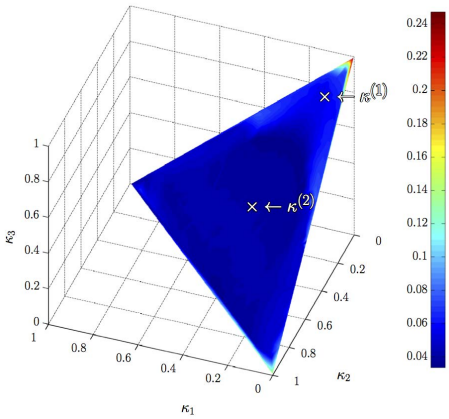
(a) Global eigenvectors



(b) Semi-supervised, with $\kappa^{(1)} = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$



(c) Semi-supervised, with $\kappa^{(2)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



(d) Test error on the κ simplex

Figure 9: The effect of varying the correlation/locality parameter κ on the classification accuracy. 9(a), 9(b), 9(c) show the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and 9(d) shows the test error as the κ vector is varied over the unit simplex.

how many nodes will be touched). Again we construct the full $70,000 \times 70,000$ k-NN graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$, where σ_i^2 is the Euclidian distance of the i^{th} node to it's nearest neighbor; and from this we define the graph Laplacian in the usual way. Using this representation we compute 3 semi-supervised eigenvectors seeding using 50 samples from each class (4s vs. 9s). However, in this case, we fix the regularization parameter

vector as $\gamma = [-0.0150, -0.0093, -\mathrm{vol}(G)]$; and note that choosing these specific values correspond to the solutions visualized in Figure 9(c) when the equations are solved exactly. Figure 10(a) shows the results for $\epsilon = 0.001$. This approximation gives us sparse solutions, and the histogram in the second row illustrates the digits that are assigned a nonzero value in the respective semi-supervised eigenvector. In particular, note that most of the mass of the eigenvector is distributed on 4s and 9s; but, for this choose of ϵ, only few digits of interest ($\approx 2.8243\%$, meaning, in particular, that not all of the 4s and 9s) have been touched by the algorithm. This results in the lack of a clean separation between the two classes as one sweeps along the leading semi-supervised eigenvector, as illustrated in the first row; the very uniform correlation distribution $\kappa = [0.8789, 0.0118, 0.1093]$; and the high classification error, as shown in the ROC curve in the bottom panel.

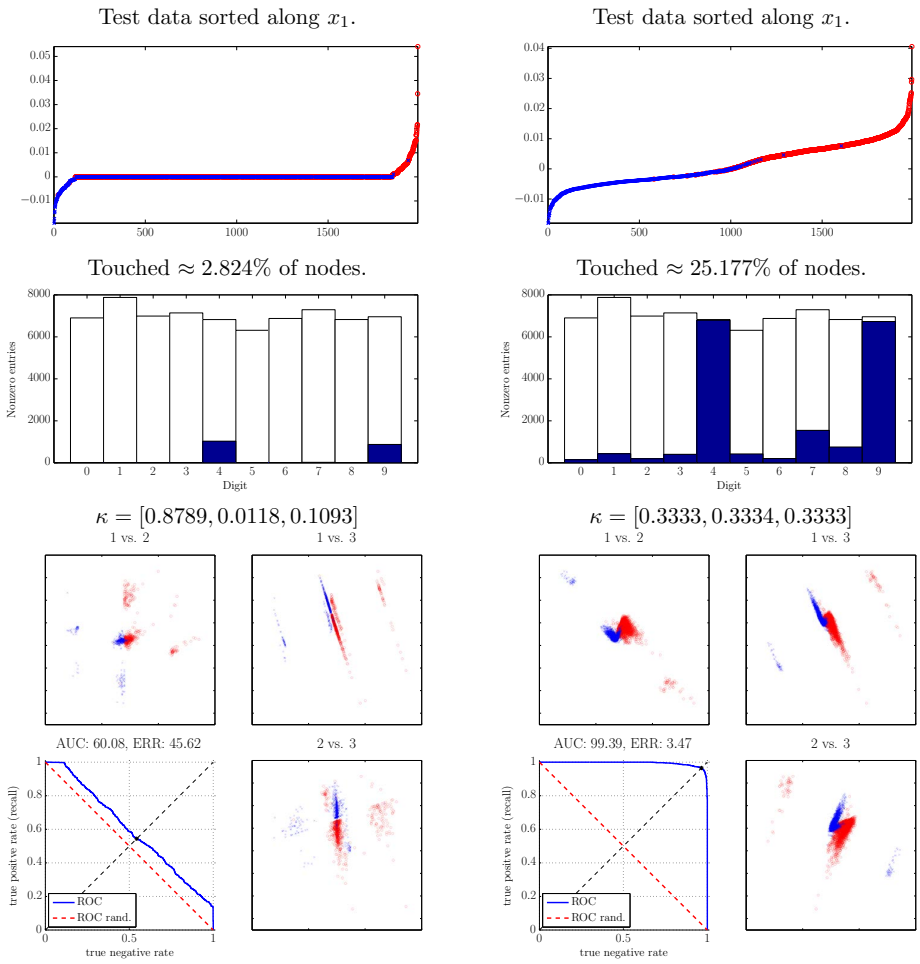
Consider, next, Figure 10(b), which shows the results for $\epsilon = 0.0001$, *i.e.*, where the locality parameter ϵ has been reduced by an order of magnitude. In this case, the algorithm reproduces the solution by touching only $\approx 25.177\%$ of the nodes in the graph, *i.e.*, basically all of the 4s and 9s and only a few other digits. This leads to a much cleaner separation between the two classes as one sweeps over the leading semi-supervised eigenvector; a much more uniform distribution over κ; and a classification accuracy that is much better and is similar to what we saw in Figure 9(c). This example illustrates that this push-peeling approximation provides a principled manner to generalize the concept of semi-supervised eigenvectors to large-scale settings, where it will be infeasible to touch all nodes of the graph.

5.3.4 Effect of low-rank Nyström approximation

Here we discuss the use of the low-rank Nyström approximation which is commonly used in large-scale kernel-based machine learning. The memory requirements for representing the explicit kernel matrix, that we here take to be our graph, scales with $\mathcal{O}(N^2)$, whereas inverting the matrix scales with $\mathcal{O}(N^3)$, which, in large-scale settings, is infeasible. The Nyström technique subsamples the dataset to approximate the kernel matrix, and the memory requirements scales with $\mathcal{O}(nN)$ and runs in $\mathcal{O}(n^2N)$, where n is size of the subsample. For completeness we include the derivation of the Nyström approximation for the normalized graph Laplacian in Appendix C.

In the beginning of Section 5.3 we constructed the $70,000 \times 70,000$ k-nearest neighbor graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$. Such a sparse construction reduces the effect of "hubs", as well as being fairly insensitive to the choice of kernel parameter, as the 10 nearest neighbors are likely to be very close in the Euclidian norm. Because the Nyström method will approximate the dense kernel matrix, the choice of kernel parameter is more important, so in the following we will consider the interplay between this parameter, as well as the rank parameter n of the Nyström approximation. Moreover, to allow us to compare a rank-n Nyström approximation with the full rank-N kernel matrix, we choose to subsample the dataset for all of the following experiments, due to the $\mathcal{O}(N^2)$ memory requirements. Thus, to provide a baseline, Figure 11 shows results based on a k-nearest neighbor graph constructed from 5% and 10% percent of the training data, where in both cases we used 10% for the test data. For both cases, when compared with the results of Figure 9(c), the classification quality is degraded, and so we emphasize that the goal of the following results are not to outperform the results reported in Figure 9(c), but to be comparable with this baseline.

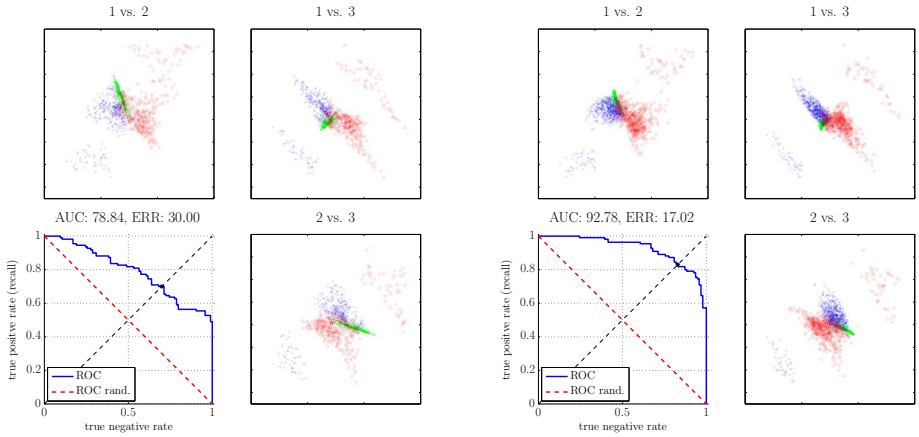
In light of this baseline, Figure 12 provides a thorough analysis for the choices of σ_i^2 that we used. Figures 12(a) and 12(b) show the classification error when using the global eigenvectors, for various rank approximations based on the Nyström method as well as the exact method (corresponding to



(a) Locality parameter $\epsilon = 0.001$

(b) Locality parameter $\epsilon = 0.0001$

Figure 10: Illustration of the push-peeling procedure to compute 3 semi-supervised eigenvectors for $\gamma = [-0.0150, -0.0093, -\mathrm{vol}(G)]$. 10(a) shows results for $\epsilon = 0.001$; and 10(b) shows results for $\epsilon = 0.0001$. First row shows the entries in the leading semi-supervised eigenvector corresponding to test points, color-coded and sorted according to magnitude; second row shows the distribution of digits touched in the full graph when executing the push algorithm; and bottom panels provide visualizations similar to the ones in Figure 9 (and shown above these is the correlation vector κ obtained for the fixed choice of γ.



(a) 5% of MNIST

(b) 10% of MNIST

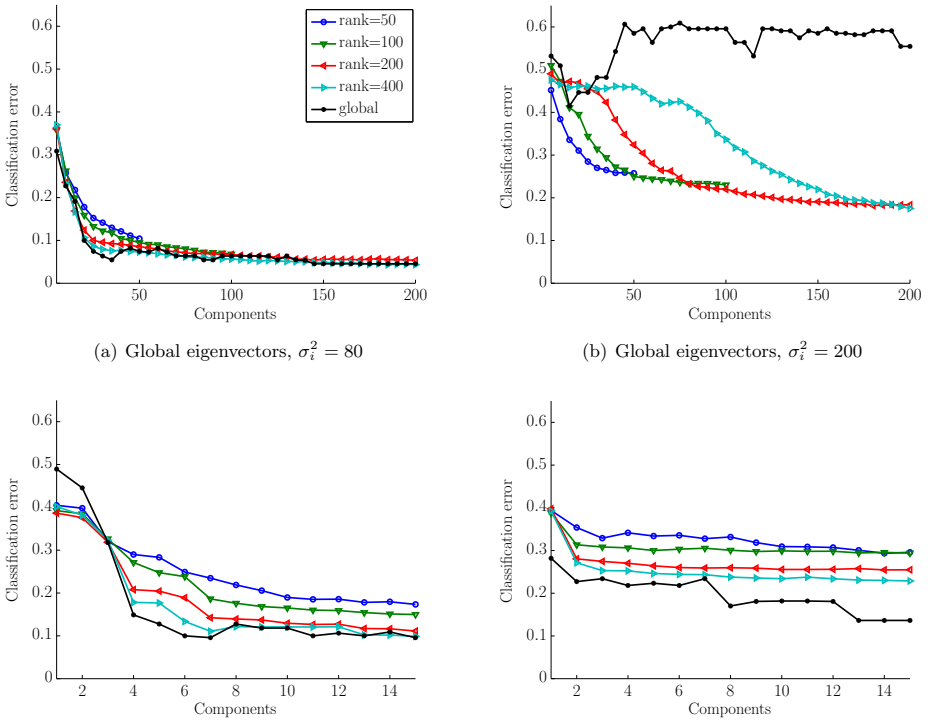
Figure 11: Example of the impact of subsampling the data set down to 5% (in 11(a)) and 10% (in 11(b)) of the original size. Remaining parameters are the same as in Figure 9(c), which shows the result to which these two plots should be compared.

rank $= n$). Interestingly, these two plots are very dissimilar in terms of their behavior as a function of the number of components. In particular, the plot in Figure 12(b) shows that the low rank approximations for a given set of components outperform the high rank approximations, and the exact representation fails to reduce the error beyond 0.4 for any of the considered set of components. This may seem counterintuitive, but the reason for this type of behavior is that the relevant global eigenvectors, for $\sigma_i^2 = 200$, are located far from the end of the spectrum (if we visualized more components for rank $= n$ the classification error would eventually drop). For the same reason, the low rank approximations improve more rapidly than the high rank approximations, as the latter approximate the lower part of the spectrum better, and these turn out to have poor discriminative properties. In contrast, the results shown in Figure 12(a) provide good class separation in the lower part of the spectrum, resulting in the high rank approximations to reduce the error most rapidly.

Finally, Figures 12(c) and 12(d) show the classification error for the SGT trained using the semi-supervised eigenvectors. (Note that the scale of the x-axis is much smaller in these subfigures.) For both kernel widths (in both Figures 12(c) and 12(d)), the ordering of the approximations are similar, *i.e.*, the semi-supervised eigenvectors constructed from the rank $= n$ approximation performs the best. Moreover, the gap between the rank $= 400$ and rank $= n$ is largest for $\sigma_i^2 = 200$, again suggesting this approximation is of insufficient rank to model the relevant local heterogeneities deep down in the spectrum; whereas for $\sigma_i^2 = 80$, the rank $= 400$ the approximation comes very close to the exact representation, suggesting that local structures are well modeled near the end of the spectrum.

To summarize these results, the method of semi-supervised eigenvectors successfully extracts relevant local structures to perform locally-biased classification, even when they are located far from the end of the spectrum. Moreover, in both cases we considered, the classification error is

reduced significantly by using only a few locally-biased components. This contrasts with the global eigenvectors, where for $\sigma_i^2 = 80$ at least 20 eigenvectors are needed in order to obtain similar performance; and for $\sigma_i^2 = 200$, the classification error remains high even for 200 eigenvectors in case of rank $= n$.



(a) Global eigenvectors, $\sigma_i^2 = 80$

(b) Global eigenvectors, $\sigma_i^2 = 200$

(c) Semi-supervised eigenvectors, $\sigma_i^2 = 80$

(d) Semi-supervised eigenvectors, $\sigma_i^2 = 200$

Figure 12: We consider 10% of the MNIST training and test data and investigate the classification accuracy of a downstream SGT classifier for various approximations of the dense similarity matrix. 12(a) and 12(b): Classification error for the SGT evaluated directly on global eigenvectors, based on various Nyström approximations and the two choices of the kernel width parameter (respectively, $\sigma_i^2 = 80$ and $\sigma_i^2 = 200$). 12(c) and 12(d): Classification error we have used the Nyström approximations as basis for computing semi-supervised eigenvectors that are then used in the downstream SGT classifier. All plots show the mean over 30 repetitions.

5.3.5 Implementation issues and running time considerations

Here, we discuss implementation details and investigate the advantage of using the Graphics Processing Unit (GPU) for computing semi-supervised eigenvectors. Although the computations under-

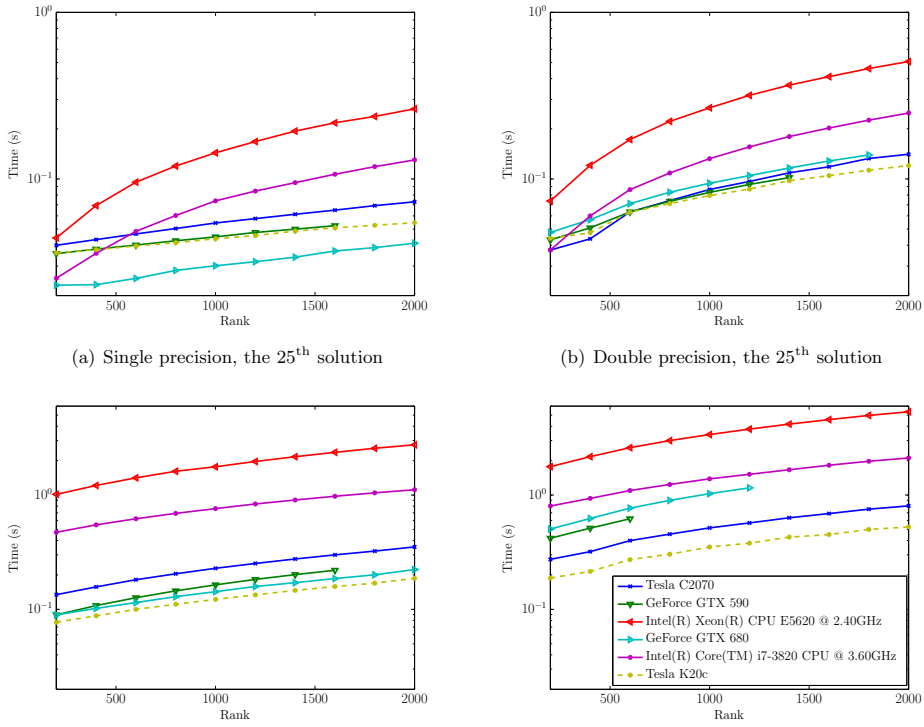
lying the construction of semi-supervised eigenvectors could be performed in many computational environments, the GPU architecture fits well with the dense semi-supervised eigenvector computation in Eqn. (9); for each component, this expression will be executed $\mathcal{O}(\log_2((\lambda_2(G) + \mathrm{vol}(G)))/\epsilon)$ times within the binary search of Algorithm 1.

Compared to a Central Processing Unit (CPU), which is well-suited for processing code with a complex control flow, a GPU is much better suited for addressing problems that can be expressed as data-parallel computations with a high arithmetic intensity [33, 9, 25]. A GPU consists of a set of Multi Processors (MPs), each containing multiple Scalar Processors (SPs), as well as different types of local memories that the SPs may access. All MPs have also access to a large global memory that, compared to their internal memories, is much slower to access. A computation task to be executed on such a device is usually setup in a grid, where each element in the grid gets assigned to a thread. The grid is then decomposed into blocks that are scheduled onto the MPs with available resources, and the assigned MP will schedule the elements of the block onto its SPs in warps with 32 threads. The best performance is obtained when all threads in a warp execute the same instruction and when the total number of threads in the grid is large, as this allows various latencies to be overlapped with arithmetic operations.

We compare most recent CPU and GPU devices in computing the solution to Eqn. (9). In terms of the GPUs we test both consumer devices (GeForce) and professional devices (Tesla), where the latter provides enhanced performance for double-precision floating point arithmetic. For a fair comparison, we decided to rely on the BLAS[3] and CUBLAS implementations as used in MATLAB 2012B, *i.e.*, avoiding to favor specific architecturally dependent implementation optimizations, since BLAS and CUBLAS should be optimal in terms of the underlying architecture. Figure 13 shows performance measures (wall-clock-time as a function of the rank parameter) of CPU and GPU experiments. For single precision arithmetic the GTX 680 scales very well, and it ends up being more than three times faster than the i7-3820, as well as noticeably faster than the previous generation high-end Tesla C2070, and it even outperforms the latest generation Tesla K20c, as seen in Figure13(a). As seen in Figure 13(a) and 13(b), the GPUs perform poorly in the low-rank regime, and this is explained by the overhead of transferring data back and forth from the main memory and to the device. However, for the high-rank matrices the arithmetic intensity increases and the overhead is less dominant. Also evident is the performance improvement of the latest CPU generation (i7-3820), that for the considered operation ends up being more than twice as fast as a previous generation E5620, that primarily is due to the higher clock frequency. For double precision arithmetic, the GTX 680 and GTX 590 are due to memory constraints stopped prematurely in the experiments, as they respectively are equipped with 2048MB and 1536MB (per GPU). Note that even though the GTX 590 is a dual GPU card, it is from the GPU computing perspective setup as two individual devices, and only one of these are used for the experiments. Interestingly the older GTX 590 outperforms the recent GTX 680, which may be explained by a higher memory bandwidth. In Figure 13(d) the Tesla K20c outperforms all other devices by a fair margin, being ≈ 1.5 times as fast as the Tesla C2070, and four times faster than the i7-3820.

Using GPU computing we are able to reduce the computation time considerably. Depending on the application of the semi-supervised eigenvectors, the advantage may be significant, for example if applied in time critical applications such as online and real-time applications or large-scale simulations.

[3]The BLAS implementation uses all physical CPU cores.



(a) Single precision, the 25^{th} solution

(b) Double precision, the 25^{th} solution

(c) Single precision, the 500^{th} solution

(d) Double precision, the 500^{th} solution

Figure 13: Running time performance measurements for solving Eqn. (9), given a specific value of the parameter γ, on the entire MNIST data set consisting of $70,000$ samples, as a function of the rank parameter. Single and double precision arithmetic results are respectively shown in 13(a) and 13(b) for the task of computing the 25^{th} solution, *i.e.*, constrained to be perpendicular to the previous 24 solutions. Similar does 13(c) and 13(d) show performance results for computing the 500^{th} solution, and here the advantage of using recent GPU architectures become even more evident, as the operation is dominated by a high arithmetic intensity that fit well with such architectures.

5.4 Functional Magnetic Resonance Imaging Data

The next dataset we consider is from functional Magnetic Resonance Imaging (fMRI). Here data analysis usually considers the characterization of relations between cognitive variables and individual brain voxels, for instance using the mass-univariate General Linear Model (GLM), where statistical parametric maps are used to identify regions of gray matter that are significantly related to particular effects under study [19]. Even though such a voxel-wise univariate approach has been tremendously productive, there are obvious limits on what can be learned about cognitive states by only examining isolated voxels [42]. Multivariate methods have therefore paved the way for

more advanced paradigms involving complex cognition, where the latent brain state cannot solely be determined from looking at individual voxel time series [16, 30, 7]. However, an immediate challenge for multivariate approaches is that weak signals carried by a sparse set of voxels can be very hard to detect, and for this reason multivariate approaches are often accompanied by spatial priors, to improve on the signal-to-noise ratio (SNR).

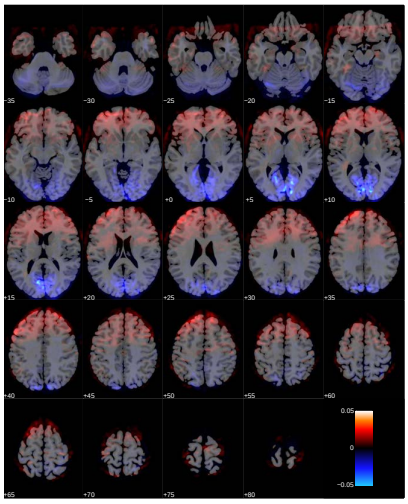
Searchlight is an algorithm that scans through the whole brain by running multiple multivariate region-of-interest (ROI) analyses, measuring the respective generalization performance, and outputs a brain map showing which regions exhibit the best discriminative properties, for example measured by classification accuracy for a particular subject task [32]. This approach was for instance applied by [27], who used it to find regions in the brain that are predictive with respect to human intentions. Compared to a univariate approach, searchlight takes advantage of the power of multivariate techniques, with the caveat that it only performs well if the target signal is available within the area covered by the ROI. This limitation is indeed shared by the univariate approaches, but with searchlight we have the freedom to increase or decrease the ROI, depending on the structure of the considered problem. If the ROI is small we approach a univariate analysis, whereas if the ROI is large, the information localization becomes less specific. Thus, if the multivariate signal is spatially distributed the searchlight approach will fall short, and simply increasing the ROI may not be a solution as irrelevant time series will decrease the SNR.

The semi-supervised eigenvectors can be used to construct a spatially-guided basis that naturally allows for spatially distributed signal representations. This strategy shares many similarities with there searchlight approach, but it is not tied to a particular ROI, and it can span distributed voxel time series that are similar in terms of our graph representation. Using the semi-supervised eigenvectors on the voxel $\times$ voxel similarity graph in this way will yield a low dimensional representation that we can project the fMRI voxel time series onto, and in that projected space we can apply any suitable classification algorithm.

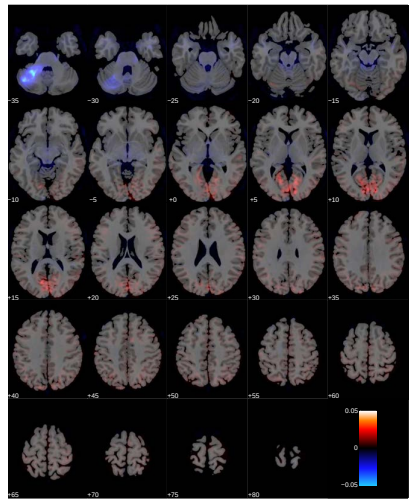
We tested the method on Blood Oxygenation Level Dependent (BOLD) sensitive fMRI data acquired on a 3 Tesla MR scanner (Siemens Magnetom Verio). Additional sequence parameter were as follows: 25 interleaved echo planar imaging gradient echo slices, echo time 30 ms, repetition time 1390 ms, flip angle 90 degrees. During the scanning session (1300 volumes) the subject was engaged in a simple motor paradigm in which the subject was asked to respond with key-presses when a visual cue was presented, and the classification task is to detect such key-presses. Pre-processing steps included: rigid body realignment, spatial smoothing (6 mm full width at half maximum isotropic Gaussian kernel), and high-pass filtering (cut-off frequency 1/128 Hz). See [56] for more details.

We construct a voxel $\times$ voxel 10-nearest neighbor graph using the nonlinear affinity $w_{ij} = \exp(-\|z_i - z_j\|^2)$. Figure 14 shows the 4 leading non-trivial global eigenvectors projected onto a sliced brain. Note that the first slice (top left) in such an image corresponds to the bottom of the brain, whereas the last slice (bottom right) corresponds to the top of the brain. The non-trivial global eigenvectors aim to span the most dominant sources of variation in the data, which in this particular dataset appears to stem mainly from the primary visual cortex (V1), and a frontal/posterior contrast apparent in the second global eigenvector. Importantly, the global eigenvectors are typically not associated with the interesting features of the task but rather general signal variation, which may be due to visual presentation of the stimuli (visual cortex) and often physiological noise sources typically dominant in the lower slices of the brain near large arteries.

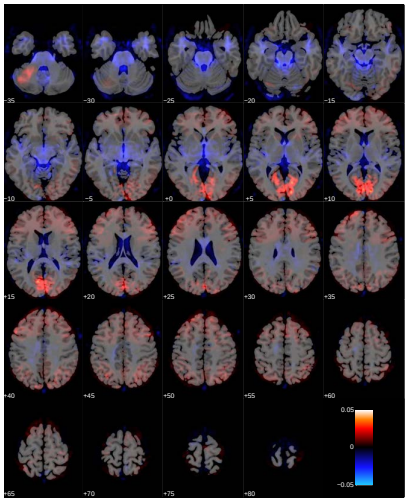
Using a probabilistic functional atlas created by averaging across multiple subjects [17], we



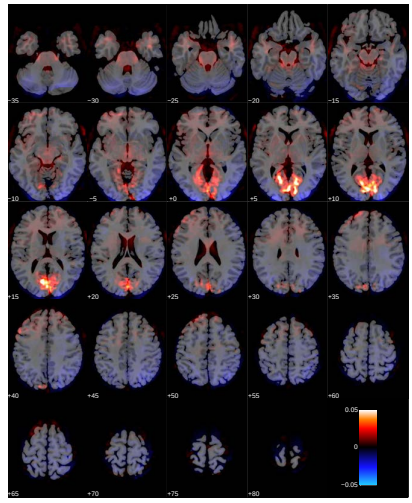
(a) Global eigenvector, v_2



(b) Global eigenvector, v_3



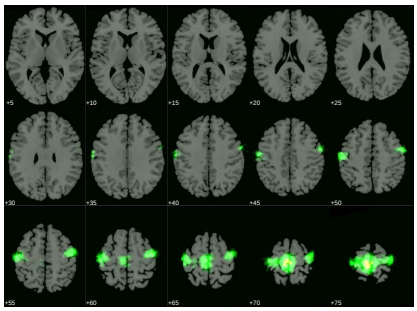
(c) Global eigenvector, v_4



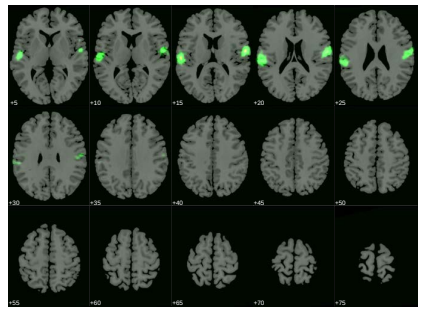
(d) Global eigenvector, v_5

Figure 14: Visualization of the leading 4 nontrivial global eigenvectors.

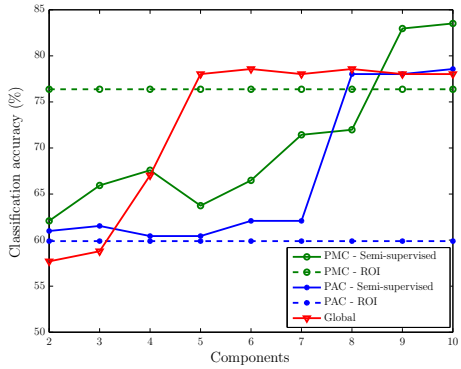
carry out two experiments based on semi-supervised eigenvectors. Specifically, we construct semi-supervised eigenvectors seeded in Primary Motor Cortex (PMC), known to be highly involved in the subject task [21], as well as semi-supervised eigenvectors seeded in Primary Auditory Cortex (PAC), that is not expected to carry much signal with respect to our target variable [39]. The seed



(a) Primary Motor Cortex (PMC)



(b) Primary Auditory Cortex (PAC)

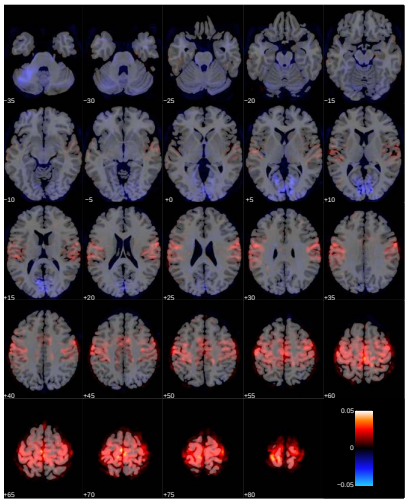


(c) Classification accuracy

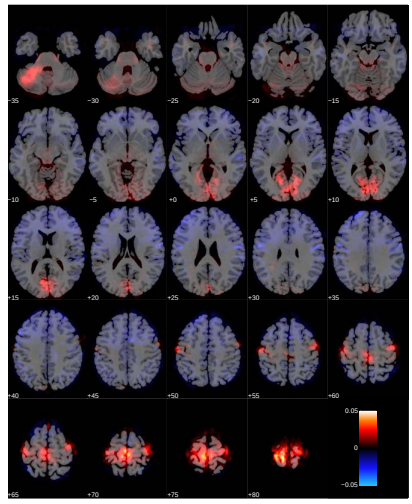
Figure 15: Figure 15(a) shows the seed region in PMC, and Figure 15(b) shows the seed region in PAC. The plot in Figure 15(c) shows the classification accuracy for the 5 different features extraction approaches. The dashed lines mark the reference where all voxel time series, as covered by the seed, are used in the downstream classifier, and the solid ones correspond to the accuracy obtained from projecting the data onto the semi-supervised eigenvectors seeded in PAC and PMC, as well as the global eigenvectors.

regions are highlighted in Figure 15(a) and 15(b).

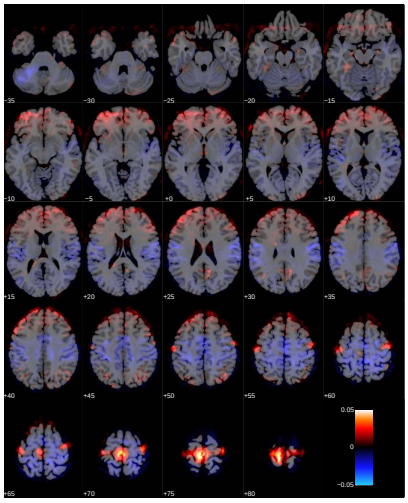
Figure 16 and 17 shows respectively the leading 4 semi-supervised eigenvectors, each having a correlation of 0.25 with the seed, and respectively seeded in PMC and PAC. As expected the semi-supervised eigenvectors are dominant near the seed region but are able to spread to related regions which carry information about important signal variation. For the PAC seed the first eigenvector appears to capture the general pattern of signal variation in part of the cortex that focus on auditory processing. The remaining three eigenvectors appear to span specific signal variations in the PAC that are more specific to subregions with the auditory cortex.



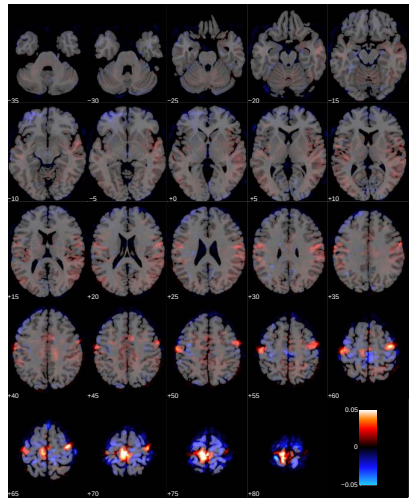
(a) Semi-supervised eigenvector (PMC), x_1



(b) Semi-supervised eigenvectors (PMC), x_2



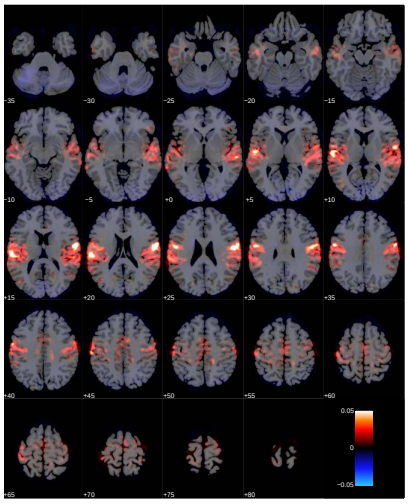
(c) Semi-supervised eigenvectors (PMC), x_3



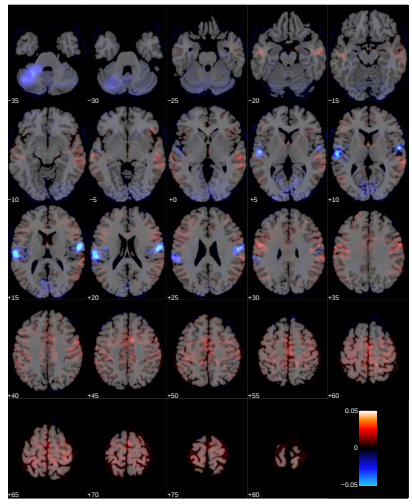
(d) Semi-supervised eigenvectors (PMC), x_4

Figure 16: Visualization of the leading 4 semi-supervised eigenvectors seeded in PMC, each correlating 0.25 with the seed, that is visualized in Figure 15(a).

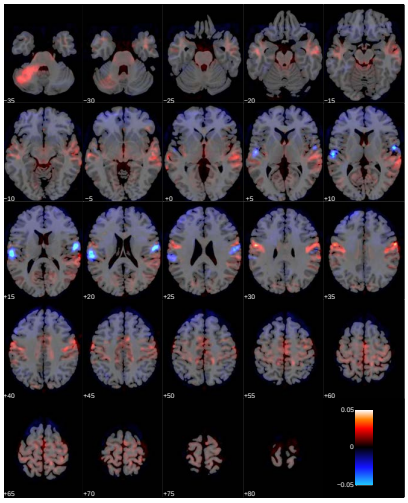
Likewise the first semi-supervised eigenvector from the seed in the PMC reveals other dominant parts of the motor network including the remaining parts of the PMC (posterior part of Brodmann area 4), somatosensory cortex (Brodmann areas 1,2 and 3) and the premotor cortex (Brodmann



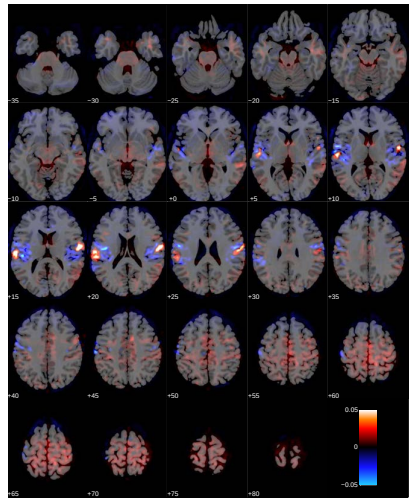
(a) Semi-supervised eigenvector (PAC), x_1



(b) Semi-supervised eigenvectors (PAC), x_2



(c) Semi-supervised eigenvectors (PAC), x_3



(d) Semi-supervised eigenvectors (PAC), x_4

Figure 17: Visualization of the leading 4 semi-supervised eigenvectors seeded in PAC, each correlating 0.25 with the seed, that is visualized in Figure 15(b).

area 5). The remaining semi-supervised eigenvectors again focus on more localized sources of signal within these areas as well as signal variation in the primary visual cortex (Brodmann area 17), which is to be expected as the visual presentation of stimuli is related to motor function in the present

task.

For comparison in our classification task, we consider the leading global eigenvectors of the graph Laplacian, as well as simply extracting the time series as specified by the seed regions. For all of the considered feature extraction approaches we use either the projected or extracted time series as data for a linear SVM that is responsible for the downstream classification task. Figure 15(c) summarizes the classification accuracies obtained by performing leave-one-out cross validation as a function of the number of components. For each semi-supervised eigenvector we fix $\kappa = \frac{1}{k}$ where k is the number of components. Hence, for two components, each correlates 0.5 with the seed, and so forth. In the same plot, the dashed blue line corresponds to classifying the brain state using only voxel time series in the region as defined by PAC. Unsurprisingly, for the dashed green line, corresponding to PMC, it is evident that the primary motor cortex is a much better proxy for predicting motor responses. Due to inter-subject variability there is no guarantee that the rigid body realignment will align the seed perfectly with the physical region, which explains why the data-driven global eigenvectors are able to yield an even higher accuracy than the PAC time series. Also seen is the "bump" in classification accuracy for the global eigenvectors, when we reach 4-5 components. Thus, for this particular dataset, relevant parts of the are signal are captured in this regime.

In the regime of few semi-supervised eigenvectors, the solutions are too localized to explain relevant local heterogeneities both near and within the seed set. As we increase the number of components they become less localized, and the semi-supervised eigenvectors seeded in PMC eventually surpasses the accuracy of global approach. As we consider more and more components, while distributing the correlation evenly across the semi-supervised eigenvectors, they will eventually converge to the global eigenvectors. Complementary, in the limit of a single component, the projection onto the leading trivial global eigenvector will simply correspond to the average time series, whereas for a leading semi-supervised eigenvector the solution is simply the seed itself, *i.e.*, the projection onto this corresponds to a weighted average in the region defined by the seed. Hence, as seen in Figure 15(c) there exists a regime in which the semi-supervised approach performs better as we are able to pickup the relevant local heterogeneities at that particular scale, given that the seed is relevant with respect to the subject task.

5.5 Large-scale Network Data

The final datasets we consider are from a collection of large sparse networks [44, 45, 46]. On these data, we demonstrate that the Push-peeling Heuristic introduced in Section 4.2 is attractive due to an improved running time, as compared to solving a system of linear equations. Moreover, we also show that the ability to obtain multiple semi-supervised eigenvectors depends on the degree heterogeneity near the seed. Finally, we empirically evaluate the influence of the ϵ parameter of the Push algorithm that implicitly determines how many nodes the algorithm will touch. This parameter can be interpreted as a regularization parameter (different from γ parameter), and setting it too large means we fail to distribute mass in the network, so that a few semi-supervised eigenvectors will consume all of the correlation. In particular, this behavior was investigated on the MNIST digits in Section 5.3.3. The basic properties for the networks considered in this section are shown in Table 2.

We start by considering the moderately sized networks from the DIMACS implementation challenge, as these networks are commonly used for the purpose of measuring realistic algorithm performance. Figure 18 shows analysis results for 6 networks from this collection, where we evaluate

the performance and feasibility of the Push algorithm for approximating the leading semi-supervised eigenvector.

Network name	Number of nodes	Number of edges
DIMACS10/de2010	24,115	116,056
DIMACS10/ct2010	67,578	336,352
DIMACS10/il2010	451,554	2,164,464
DIMACS10/smallworld	100,000	999,996
DIMACS10/333SP	3,712,815	22,217,266
DIMACS10/AS365	3,799,275	22,736,152
LAW/arabic-2005	22,744,080	1,107,806,146
LAW/indochina-2004	7,414,866	301,969,638
LAW/it-2004	41,291,594	2,054,949,894
LAW/sk-2005	50,636,154	3,620,126,660
LAW/uk-2002	18,520,486	523,574,516
LAW/uk-2005	39,459,925	1,566,054,250

Table 2: Summary of the networks considered in this section. Some of these networks are directed and have been symmetrized for the purpose of this analysis, *i.e.*, the number edges in this table refer to the number of edges in the undirected graph.

As stated in Section 3.3, diffusion based procedures such as the Push algorithm can be used to solve our objective for $\gamma < 0$. The impact of the reduced search range is that such procedures may not be able to produce a uniform correlation distribution for a set of semi-supervised eigenvectors. Hence, the leading solution(s) will instead pickup too much correlation, because sufficient mass cannot to diffuse away from the seed set. However, the effect of a non-uniform correlation distribution was analyzed on the MNIST data in Section 5.3, where we found that the performance of a downstream classifier is fairly robust to such non-uniformities, as seen by the simplex in Figure 9. Consequently, we emphasize that in a large-scale setting such side effects of diffusion based procedures is offset by the advantage of a greatly improved time complexity as compared to solving the system of linear equations, that implicitly touch every node.

For each of the 6 analyzed networks in Figure 18, we run two experiments considering different seeds, using respectively a high degree and low degree single seed node. Figure 18(a)-18(c) considers census block networks characterized by heavy-tailed degree distributions, whereas Figure 18(d)-18(f) considers more densely connected synthetic networks. For each of these 6 networks the speedup is measured by comparing with a standard conjugate gradient implementation using a tolerance of 1e-6, and we stress that this tolerance cannot be directly compared with ϵ in the Push algorithm. Moreover, we test three different settings of the ϵ parameter, and we emphasize that for $\epsilon =$ 1e-4, the Push algorithm produces a similar result as the conjugate gradient algorithm. In Figure 18 this can be seen by the red curve ($\epsilon =$ 1e-4) in the correlation decay plots (see the figure caption) being on top of the black curve (conjugate gradient).

Common for Figure 18(a)-18(c) are that low degree seed nodes yield very localized solutions for the entire range of α, opposed to the high degree nodes that all succeed in gradually reducing the correlation when α is reduced. Also, the choice of ϵ is obviously very important, *i.e.*, choosing it too large results in a solution that correlates too much with the seed, whereas choosing it too small means that we will be touching more nodes than necessary, resulting in a performance penalty. In

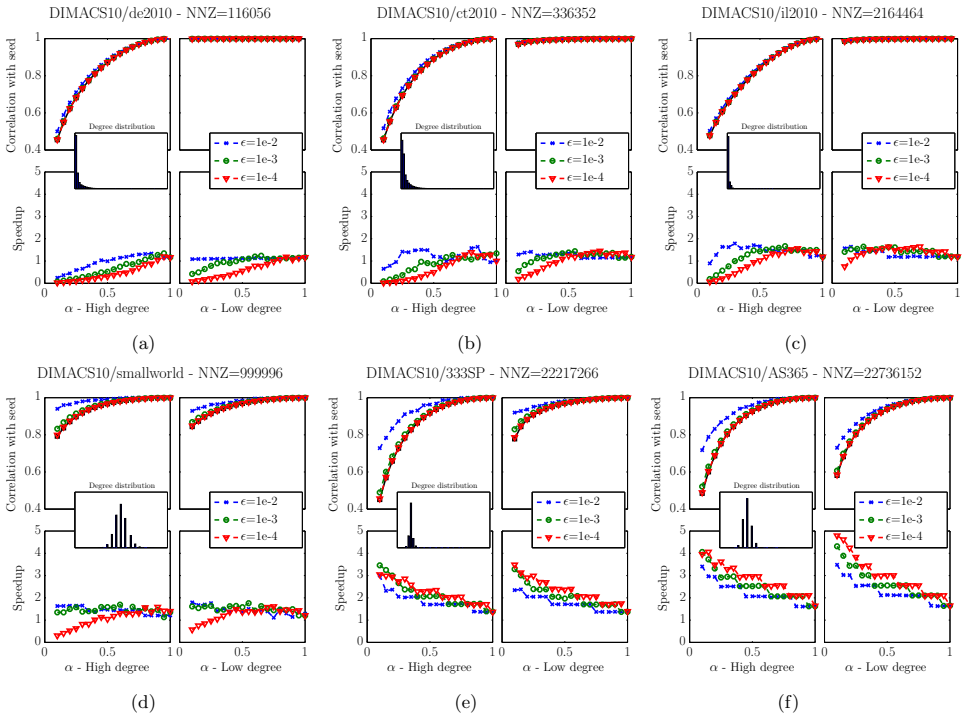


Figure 18: For each network the first row depicts how the correlation decays as α tends towards 0, whereas the bottom row shows the speedup relative to the standard approach using conjugate gradient with a tolerance of 1e-6, that is the default approach in our software distribution. Besides the three considered values of ϵ the correlation plots also illustrate the decay based on conjugate gradient (black curve), however this may be difficult to see, as the Push algorithm for ϵ = 1e-4 coincides with that solution. Finally, seeds based on a high degree and low degree node are presented in respectively the first and last column, and the degree distribution for the network is visualized in a minor overlapping plot.

general the networks analyzed in Figure 18(a)-18(c) are too small to yield significant performance improvements over the conjugate gradient algorithm, and the Push algorithm is only competitive for large values of α.

For the network in Figure 18(d), we see similar performance characteristics as the networks analyzed in Figure 18(a)-18(c) due to its small size. However, the two final networks analyzed in Figure 18(e)-18(f) share similar characteristics in terms of the degree distribution, but due to a much larger size they show significant performance improvements over the conjugate gradient algorithm. Interestingly, the Push algorithm instantiated with ϵ = 1e-4 yields a greater speedup in some settings, which may be explained by faster convergence, caused by a reduced threshold for distributing mass. Hence, the running time of the Push algorithm may not always decrease

monotonically as ϵ increases.

In general it seems that seeding in a sparsely connected region of a network results in a solution having a large correlation with the seed for most values of α. This is obviously a limiting factor if we are interested in using the peeling procedure to find multiple semi-supervised eigenvectors in that particular region. However, for large networks and more densely connected regions the benefit of the Push algorithm is immediate.

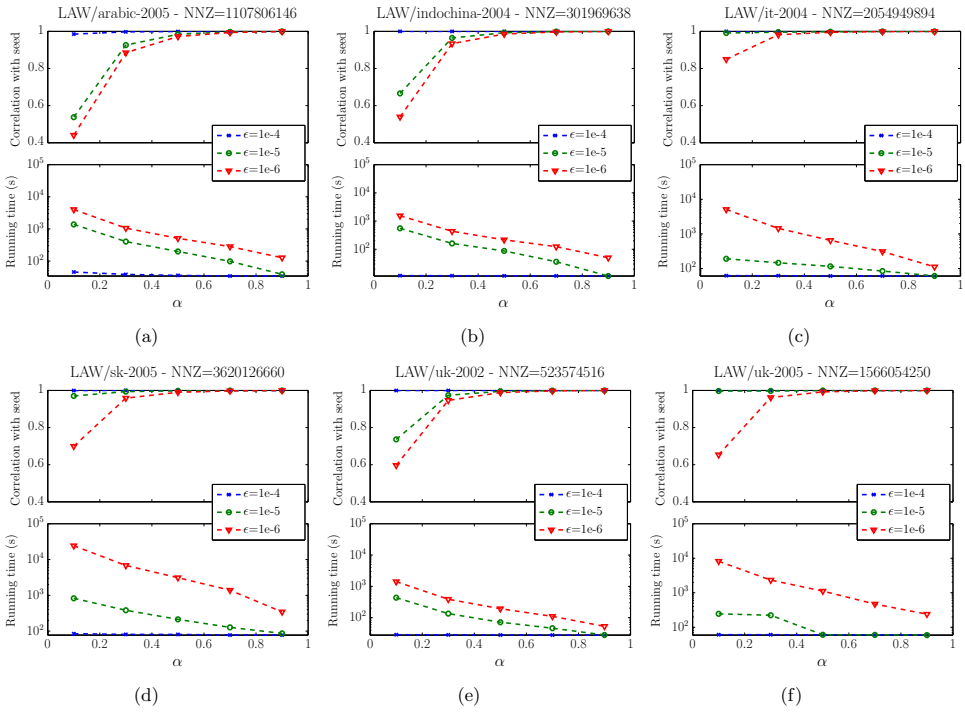


Figure 19: Visualizes results for applying the Push algorithm to 6 very large web-crawl networks. For all networks we seed in the node with the highest degree. The top plot in each subfigure shows the correlation decay as a function of α, whereas in the bottom plot we resort to absolute timings as the conjugate gradient algorithm is not feasible in this setting, as opposed to showing speedups as in Figure 18.

Finally, we scale up to demonstrate that we can adapt the notion of semi-supervised eigenvectors to large datasets, and we do so by analyzing 6 large web-crawl networks. These networks are large enough that touching all nodes is infeasible, *i.e.*, conjugate gradient is not a feasible option, so in Figure 19 we resort to absolute timings. For the analysis results shown in Figure 19, we are solely interested in giving the reader some intuition about the running time in a large-scale setting, as well as an idea on how the parameters interplay. Hence, we only consider experiments where we seed in a high degree node, as these are likely yield the worst running times, but also succeed in

reducing the correlation the most. This will make the peeling procedure described in Section 4.2 applicable, allowing us to obtain multiple semi-supervised eigenvectors. As seen for all networks analyzed in Figure 19(a)-19(f) the solution is highly sensitive to the choice of ϵ, but for all networks we are able to reduce the correlation when α tends towards 0 in case of ϵ = 1e-6. We emphasize that the reason for ϵ being smaller for these experiments, as compared to the previous is that the seed is normalized to have unit norm, implicitly requiring a lower ϵ when the network increases in size.

For diffusion based procedures to be useful with respect to the computation of semi-supervised eigenvectors, mass must be able "bleed" away from the seed set and into the surrounding network. Otherwise only few semi-supervised eigenvectors can be found as the leading solution(s) become too correlated with the seed set. For moderately sized problems conjugate gradient performs very well, but in a large-scale setting, as considered here, the presented approach proves very efficient, allowing us to compute approximations to semi-supervised eigenvectors in networks consuming more than 30GB of working memory. Obtaining an improved understanding of how the method of semi-supervised eigenvectors can be used to perform common machine learning tasks on graphs of that size is an obvious direction raised by our work.

6 Conclusion

We have introduced the concept of semi-supervised eigenvectors as local analogues of the global eigenvectors of a graph Laplacian that have proven so useful in a wide range of machine learning and data analysis applications. These vectors are biased toward prespecified local regions of interest in a large data graph; and we have shown that since they inherit many of the nice properties of the usual global eigenvectors, except in a locally-biased context, they can be used to perform locally-biased machine learning. The basic method is conceptually simple and involves solving a sequence of linear equation problems; we have also presented several extensions of the basic method that have improved scaling properties; and we have illustrated the behavior of the method. Due to the speed, simplicity, stability, and intuitive appeal of the method, as well as the range of applications in which local regions of a large data set are of interest, we expect that the method of semi-supervised eigenvectors can prove useful in a wide range of machine learning and data analysis applications.

Ackowledgements

We acknowledge Kristoffer H. Madsen, researcher at the Danish Research Centre for Magnetic Resonance at the University Hospital in Hvidovre, for providing the analyzed fMRI data.

References

[1] R. Andersen, F.R.K. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 475–486, 2006.

[2] R. Andersen and K. Lang. Communities from seed sets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 223–232, 2006.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56:209–239, 2004.

[5] M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[6] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[7] S. Bode and J. D. Haynes. Decoding sequential stages of task preparation in the human brain. *NeuroImage*, 45(2):606–13, 2009.

[8] P. Boldi and S. Vigna. The push algorithm for spectral ranking. Technical report. Preprint: arXiv:1109.4680 (2011).

[9] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 917–924. ACM, 2003.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[11] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Annual Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, pages 585–592, 2003.

[12] F.R.K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

[13] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition in data: Diffusion maps. *Proc. Natl. Acad. Sci. USA*, 102(21):7426–7431, 2005.

[14] M. Cucuringu and M. W. Mahoney. Localization on low-order eigenvectors of data matrices. Technical report. Preprint: arXiv:1109.1355 (2011).

[15] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[16] E. Eger, J. Ashburner, J. D. Haynes, R. J. Dolan, and G. Rees. fMRI activity patterns in human LOC carry information about object exemplars within category. *Journal of Cognitive Neuroscience*, 20(2):356–370, February 2008.

[17] S. B. Eickhoff, K. E. Stephan, H. Mohlberg, C. Grefkes, G. R. Fink, K. Amunts, and K. Zilles. A new SPM toolbox for combining probabilistic cytoarchitectonic maps and functional imaging data. *NeuroImage*, 25(4):1325–35, May 2005.

[18] A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, 2007.

[19] K. J. Friston, a. P. Holmes, K. J. Worsley, J.-P. Poline, C. D. Frith, and R. S. J. Frackowiak. Statistical parametric maps in functional imaging: A general linear approach. *Human Brain Mapping*, 2(4):189–210, 1994.

[20] W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114/115:815–839, 1989.

[21] S. Geyer, A. Ledberg, A. Schleicher, S. Kinomura, T. Schormann, U. Bürgel, T. Klingberg, J. Larsson, K. Zilles, and P. E. Roland. Two different areas within the primary motor cortex of man. *Nature*, 1996.

[22] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. Technical report. Preprint arXiv:1303.1849 (2012).

[23] J. Ham, D.D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, pages 000–000, 2004.

[24] T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for locally-biased learning. In *Annual Advances in Neural Information Processing Systems 25: Proceedings of the 2012 Conference*, 2012.

[25] T. J. Hansen, M. Morup, and L. K. Hansen. Non-parametric co-clustering of large scale sparse bipartite networks on the GPU. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

[26] T.H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

[27] J. D. Haynes, K. Sakai, G. Rees, S. Gilbert, C. Frith, and R. E. Passingham. Reading hidden intentions in the human brain. *Current biology : CB*, 17(4):323–8, 2007.

[28] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, 2003.

[29] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 290–297, 2003.

[30] Y. Kamitani and F. Tong. Decoding the visual and subjective contents of the human brain. *Nature neuroscience*, 8(5):679–85, 2005.

[31] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[32] N. Kriegeskorte, R. Goebel, and P. Bandettini. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the United States of America*, 103(10):3863–8, March 2006.

[33] J .Krüger and R. Westermann. Linear algebra operators for GPU implementation of numerical algorithms. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 908–916. ACM, 2003.

[34] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[35] J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 695–704, 2008.

[36] M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 121–128, 2011.

[37] M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13:2339–2365, 2012.

[38] S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2057–2064, 2011.

[39] P. Morosan, J. Rademacher, A. Schleicher, K. Amunts, T. Schormann, and K. Zilles. Human primary auditory cortex: cytoarchitectonic subdivisions and mapping into a spatial reference system. *NeuroImage*, 13(4):684–701, 2001.

[40] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[41] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS '01: Proceedings of the 15th Annual Conference on Advances in Neural Information Processing Systems*, 2001.

[42] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430, 2006.

[43] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[44] B. Paolo, C. Bruno, S. Massimo, and V. Sebastiano. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

[45] B. Paolo, R. Marco, S. Massimo, and V. Sebastiano. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*. ACM Press, 2011.

[46] B. Paolo and V. Sebastiano. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[47] P. O. Perry and M. W. Mahoney. Regularized Laplacian estimation and fast eigenvector approximation. In *Annual Advances in Neural Information Processing Systems 25: Proceedings of the 2011 Conference*, 2011.

[48] K.T. Poole. The decline and rise of party polarization in congress during the twentieth century. *Extensions: A Journal of the Carl Albert Congressional Research and Studies Center*, Fall 2005.

[49] K.T. Poole and H. Rosenthal. Patterns of congressional voting. *American Journal of Political Science*, 35:228–278, 1991.

[50] A. Pothen, H.D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[51] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

[52] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. In O. Chapelle, B. Schoelkopf, and A. Zien, editors, *Semisupervised Learning*, pages 293–308. MIT Press, 2006.

[53] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[54] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transcations of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[55] D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proceedings of the 36th annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

[56] S. C. Strother. Evaluating fMRI preprocessing pipelines. *Engineering in Medicine and Biology Magazine, IEEE*, 25(2):27–41, 2006.

[57] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Annual Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 945–952, 2002.

[58] A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

[59] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[60] S.-H. Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, pages 2–14, 2010.

[61] S. Vigna. Spectral ranking. Technical report. Preprint: arXiv:0912.0238 (2009).

[62] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[63] A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter. Party polarization in congress: A network science approach. Technical report. Preprint: arXiv:0907.3509 (2009).

[64] C.K.I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166, 2000.

[65] C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Annual Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.

[66] S. X. Yu and J. Shi. Grouping with bias. In *Annual Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 1327–1334, 2002.

[67] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Annual Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, pages 321–328, 2004.

A Supplementary Proofs

Lemma 1 *Given an SPSD matrix M and some vector x where $x^\top x = 1$, it holds that*

$$\lim_{\omega\to\infty}\left(M+\omega xx^\top\right)^+ = \left(\left(I-xx^\top\right)M\left(I-xx^\top\right)\right)^+. \tag{14}$$

Proof: Prior to applying the pseudo inverse, x is clearly an eigenvector with eigenvalue $\lambda = 0$ on the right hand side, and for left hand side x is an eigenvector with eigenvalue $\lambda = \infty$. Hence, without loss of generalizability we can decompose $M = \alpha xx^\top + X_\perp \Lambda X_\perp^\top$, where Λ is a diagonal matrix, such that $M^+ = \frac{1}{\alpha}xx^\top + X_\perp\Lambda^+X_\perp^\top$. First we consider the expansion of the left hand side of Eqn. (14)

$$\lim_{\omega\to\infty}\left(\left(\alpha+\omega\right)xx^\top + X_\perp\Lambda X_\perp^\top\right)^+ = \lim_{\omega\to\infty}\frac{1}{\alpha+\omega}xx^\top + X_\perp\Lambda^+X_\perp^\top = X_\perp\Lambda^+X_\perp^\top.$$

Similar, by expanding the right hand side we get

$$\left(\left(I-xx^\top\right)\left(\alpha xx^\top + X_\perp\Lambda X_\perp^\top\right)\left(I-xx^\top\right)\right)^+ = \left(X_\perp\Lambda X_\perp^\top\right)^+ = X_\perp\Lambda^+X_\perp^\top.$$

◇

Lemma 2 *For $M' = M + \omega\sum_i x_i x_i^\top$ where $\omega \geq 0$ it holds that $\lambda_k(M') \geq \lambda_k(M)$.*

Proof: All eigenvalues of the sum of rank-1 perturbations are non-negative

$$\omega\sum_i x_i x_i^\top \succeq 0 \Rightarrow M' \succeq M.$$

◇

Lemma 3 *Given an orthonormal basis, $X = [x_1, \ldots, x_{n-1}]$,* i.e.*, $X^\top D_G X = I$, and unit length seed $s^\top D_G s = 1$. Then, any unit length vector $x_n^\top D_G x_n = 1$, perpendicular to the subspace $X^\top D_G x_n = 0$, will have a correlation with the seed bounded by*

$$0 \leq (x_n^\top D_G s)^2 \leq 1 - \sum_{i=1}^{n-1}(x_i^\top D_G s)^2.$$

Proof: The proof follows directly from the Pythagorean theorem. Let $X = [x_1, \ldots, x_N]$ be the orthonormal basis of $\mathbb{R}^N$, *i.e.*, spanning s. Then

$$\sum_{i=1}^{N}(x_i^\top D_G s)^2 = (s^\top D_G s)^2 = 1.$$

◇

Lemma 4 *For the matrix $P_\gamma = \mathcal{L}_G - \gamma I$ it holds that*

$$P_\gamma^+ - P_{\hat{\gamma}}^+ = (\gamma - \hat{\gamma}) P_{\hat{\gamma}}^+ P_\gamma^+, \tag{15}$$

given that neither γ nor $\hat{\gamma}$ coincides with an eigenvalue of $\mathcal{L}_G$.

Proof: The proof follows directly by plain algebra. Simply substitute the SVD $P_\gamma = V \Lambda_\gamma V^T$, where Λ_γ is a diagonal matrix with the eigenvalues shifted by γ, into Eqn. (15)

$$\begin{aligned} V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ V^T V \Lambda_\gamma^+ V^T \\ V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+ V^T \\ \Rightarrow \Lambda_\gamma^+ - \Lambda_{\hat{\gamma}}^+ &= (\gamma - \hat{\gamma}) \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+. \end{aligned}$$

The system is decoupled so it will be sufficient to consider a single eigenvalue

$$\begin{aligned} \frac{1}{\lambda_i - \gamma} - \frac{1}{\lambda_i - \hat{\gamma}} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\lambda_i - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} - \frac{\lambda_i - \gamma}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)}. \end{aligned}$$

Also, this trivially holds for the rank deficient case, *i.e.*, $0 = 0$.

$\diamond$

Lemma 5 *As pointed out in Section 3.3, it is already immediate that the initial semi-supervised eigenvector can be computed using a diffusion-based procedure, such as the Push algorithm. However, from that discussion it remains unclear how the approach can be generalized for the consecutive $k-1$ semi-supervised eigenvectors. It turns out that the k^{th} solution is approximated by*

$$x_k \approx c(I - XX^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \tag{16}$$

given that $(L_G - \gamma_k D_G)^+ D_G s$ is linearly independent with respect to the previous $k-1$ solutions contained in X.

Proof: By Eqn. (9) the solution for the second semi-supervised eigenvector can be expressed as

$$y_2 = c\left({P_{\gamma_2}}^+ - {P_{\gamma_2}}^+ y_1 (y_1^T {P_{\gamma_2}}^+ y_1)^+ y_1^T {P_{\gamma_2}}^+\right) D_G^{1/2} s,$$

where $(y_1^T {P_{\gamma_2}}^+ y_1)^+$ is a constant. For notational convenience we start by substituting $b = D_G^{1/2} s$ together with the explicit solution $y_1 \propto {P_{\gamma_1}}^+ b$

$$y_2 = c{P_{\gamma_2}}^+ b - \frac{c{P_{\gamma_2}}^+ {P_{\gamma_1}}^+ b b^T {P_{\gamma_1}}^+ {P_{\gamma_2}}^+ b}{b^T {P_{\gamma_1}}^+ {P_{\gamma_2}}^+ {P_{\gamma_1}}^+ b},$$

and for the same reason we also introduce $\rho_{\gamma_1\gamma_2} = b^\top {P_{\gamma_1}}^+ {P_{\gamma_2}}^+ b$

$$y_2 = c{P_{\gamma_2}}^+ b - \frac{c\rho_{\gamma_1\gamma_2} {P_{\gamma_2}}^+ {P_{\gamma_1}}^+ b}{b^T {P_{\gamma_1}}^+ {P_{\gamma_2}}^+ {P_{\gamma_1}}^+ b}.$$

We can approximate this expression by exploiting the structural result of Lemma 4, namely that ${P_{\gamma_1}}^+ - {P_{\gamma_2}}^+ = (\gamma_1 - \gamma_2){P_{\gamma_2}}^+{P_{\gamma_1}}^+$

$$\begin{aligned} y_2 &\approx c{P_{\gamma_2}}^+ b - \frac{c\rho_{\gamma_1\gamma_2}({P_{\gamma_1}}^+ - {P_{\gamma_2}}^+)b}{b^T{P_{\gamma_1}}^+({P_{\gamma_1}}^+ - {P_{\gamma_2}}^+)b} \\ &= c{P_{\gamma_2}}^+ b - \frac{c\rho_{\gamma_1\gamma_2}({P_{\gamma_1}}^+ - {P_{\gamma_2}}^+)b}{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}. \end{aligned}$$

We emphasize that this approximation is exact whenever ${P_{\gamma_1}}^+ - {P_{\gamma_2}}^+$ is well-conditioned, and singular for $\gamma_1 = \gamma_2$. Then, substitute $c = \frac{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}{\rho_{\gamma_1\gamma_1}}$

$$\begin{aligned} y_2 &\approx \frac{\rho_{\gamma_1\gamma_1}{P_{\gamma_2}}^+ b - \rho_{\gamma_1\gamma_2}{P_{\gamma_2}}^+ b}{\rho_{\gamma_1\gamma_1}} - \frac{\rho_{\gamma_1\gamma_2}({P_{\gamma_1}}^+ - {P_{\gamma_2}}^+)b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1}{P_{\gamma_2}}^+ b - \rho_{\gamma_1\gamma_2}{P_{\gamma_2}}^+ b - \rho_{\gamma_1\gamma_2}{P_{\gamma_1}}^+ b + \rho_{\gamma_1\gamma_2}{P_{\gamma_2}}^+ b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1}{P_{\gamma_2}}^+ b - \rho_{\gamma_1\gamma_2}{P_{\gamma_1}}^+ b}{\rho_{\gamma_1\gamma_1}} \\ &= {P_{\gamma_2}}^+ b - \frac{\rho_{\gamma_1\gamma_2}{P_{\gamma_1}}^+ b}{\rho_{\gamma_1\gamma_1}}. \end{aligned}$$

By resubstituting for the auxiliary variables we obtain the desired result

$$y_2 \approx c(I - y_1 y_1^T)(\mathcal{L}_G - \gamma I)^+ D_G^{1/2} s,$$

and by applying this procedure recursively it follows that

$$y_k \approx c(I - YY^T)(\mathcal{L}_G - \gamma_k I)^+ D_G^{1/2} s.$$

Finally, we can relate this result to the combinatorial graph Laplacian as follows

$$\begin{aligned} y_k &\approx c(I - D_G^{1/2} X X^T D_G^{1/2}) D_G^{1/2} (L_G - \gamma_k D_G)^+ D_G s \\ &= c(D_G^{1/2} - D_G^{1/2} X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s \\ &= c D_G^{1/2} (I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \end{aligned}$$

and due to the relationship $x_k = D_G^{-1/2} y_k$ it follows that

$$x_k \approx c(I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s.$$

$\diamond$

B Derivation of sparse graph diffusions.

To allow efficient computation of semi-supervised eigenvectors by graph diffusions, we must make the relationship with the sparse seed vector explicit. Here we specifically consider the derivation of Eqn. (13). Given a sparse seed indicator s_0, we can write the seed vector s as $s \propto D_G^{-1/2}(I - v_0 v_0^T)s_0$, where $v_0 \propto \mathrm{diag}(D^{1/2})$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-one vector of the combinatorial graph Laplacian). Using this explicit form of s we can rewrite the leading solution as

$$\begin{aligned} x_1 &= c(L_G - \gamma D_G)^+ D_G s \\ &= c D_G^{-1/2}(\mathcal{L}_G - \gamma I)^+ D_G^{1/2} s \\ &= c D_G^{-1/2}(\mathcal{L}_G - \gamma I)^+ D_G^{1/2} D_G^{-1/2}(I - v_0 v_0^T)s_0 \\ &= c D_G^{-1/2}\left((\mathcal{L}_G - \gamma I)^+ s_0 - (\mathcal{L}_G - \gamma I)^+ v_0 v_0^T s_0\right). \end{aligned}$$

Since $\mathcal{L}_G - \gamma I$ simply shifts the eigenvalues of $\mathcal{L}_G$ by $-\gamma$, the latter term simplifies to

$$\begin{aligned} x_1 &= c D_G^{-1/2}\left((\mathcal{L}_G - \gamma I)^+ s_0 - \left(\frac{1}{-\gamma} v_0 v_0^T\right) v_0 v_0^T s_0\right) \\ &= c D_G^{-1/2}\left((\mathcal{L}_G - \gamma I)^+ s_0 + \frac{1}{\gamma} v_0 v_0^T s_0\right) \\ &= c D_G^{-1/2}\left(\frac{1}{-\gamma} D_G^{-1/2} \mathrm{pr}_\epsilon\left(\frac{\gamma}{\gamma - 2}, D_G^{1/2} s_0\right) + \frac{1}{\gamma} v_0 v_0^T s_0\right). \end{aligned}$$

Finally, by exploiting the peeling result in Eqn. (16), we can use the Push algorithm to approximate the sequence of semi-supervised eigenvectors in an extremely efficient manner

$$x_t^* \approx c\left(I - X X^T D_G\right)\left(D_G^{-1} \mathrm{pr}_\epsilon\left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2} s_0\right) - D_G^{-1/2} v_0 v_0^T s_0\right),$$

as the Push algorithm is only applied on the sparse seed set.

C Nyström Approximation for the Normalized Graph Laplacian

The vanilla procedure is as follows; we choose m samples at random from the full data set, and for notational simplicity we reorder the samples so that these m samples are followed by the remaining $n = N - m$ samples, *i.e.*, we can partition the adjacency matrix as

$$A_G = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{n \times n}$, with $N = m + n$ and $m \ll n$. The Nyström extension then approximates the huge C matrix in terms of A and B, so the resulting approximation to weight matrix becomes

$$A_G \approx \hat{A}_G = \begin{pmatrix} A & B \\ B^T & B^T A^{-1} B \end{pmatrix}.$$

Hence, rather than encoding only each nodes k-nearest-neighbors into the weight matrix, the Nyström methods provides a low-rank approximation to the entire dense weight matrix. Since the leading eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ correspond to the smallest of $\mathcal{L}_G$, our goal is to diagonalize $D_G^{-1/2} A_G D_G^{-1/2}$. At the risk of washing out the local hetrogeneties the Nyström procedure approximates the largest eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ using the normalized matrices $\tilde{A}$ and $\tilde{B}$

$$\tilde{A}_{ij} = \frac{A_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}}, \quad i,j = 1,\dots m$$
$$\tilde{B}_{ij} = \frac{B_{ij}}{\sqrt{\hat{d}_i \hat{d}_{j+m}}}, \quad i = 1,\dots m, \quad j = 1,\dots,n.$$

Finally, let $U\Lambda U^T$ be the SVD of $\tilde{A} + \tilde{A}^{-1/2}\tilde{B}\tilde{B}^T\tilde{A}^{-1/2}$, then the m leading eigenvectors are approximated by

$$V = \begin{pmatrix} \tilde{A} \\ \tilde{B}^T \end{pmatrix} \tilde{A}^{-1/2} U \Lambda^{-1/2},$$

and the normalized graph Laplacian by $\mathcal{L}_G \approx I - V\Lambda V^T$.

Appendix G

Personalized Audio Systems - a Bayesian Approach

Work in progress; will be submitted to *Audio Engineering Society (AES 2013)*

Audio Engineering Society

Convention Paper

Presented at the 135th Convention
2013 New York, United States of America

Personalized Audio Systems - a Bayesian Approach

Jens Brehm Nielsen[1,2], Bjørn Sand Jensen[1], and Toke Jansen Hansen[1]

[1] *Technical University of Denmark, DTU Compute, 2800 Lyngby, Denmark*

[2] *Widex A/S, Nymøllevej 6, 3540 Lynge, Denmark*

Correspondence should be addressed to Jens Brehm Nielsen (jenb@dtu.dk)

ABSTRACT

Modern audio systems are typically equipped with several user-adjustable parameters unfamiliar to most listeners. Nevertheless, the user is forced to perform high-dimensional optimization with respect to the user's own preference using simple trial and error methods in order to obtain optimal system settings. This possibly leads to extensive experimentation involving several tiresome adjustments-listening trials and often results in suboptimal settings. To address this problem, this paper presents a simple yet general framework for robust and intuitive personalization of audio systems with several (in principle infinite) free system parameters. The framework is based on a Bayesian viewpoint building on Gaussian process regression with corresponding sequential design methods for effectively selecting the next system configuration to be evaluated. We demonstrate the framework in a real interactive loop, where twenty-four tests subjects are given a personalized setting of a five-band constant-Q equalizer with thousands of possible settings. It shows that the proposed preference modeling approach combined with sequential design is able to find a significantly better solution than obtained with random experimentation in under the same model.

1. INTRODUCTION

The ever increasing number of features and processing possibilities in many modern multimedia systems, such as personal computers, mobile phones, hearing aids and home entertainment systems, has made it possible for users to customize these devices significantly. A downside in this trend is the large number of user adjustable parameters which makes it a daunting and complex task to actually adjust/optimize the devices correctly. For audio systems, the optimization is further complicated by perceptual and cognitive aspects of the human auditory and cognitive system, which results in a significant spread in subject's opinions concerning the adjustment of a particular device. As a consequence, users often have to navigate in a high-dimensional parame-

ter space, which makes it extremely difficult for users to find even a local optimum. It is therefore of great interest to find and evaluate fast and flexible tools for optimizing user adjustable settings with the aim to realize truly personalized audio systems.

We address this problem of optimizing settings that span a high-dimensional space and for which the preference typically induces a non-convex functions over the setting space, by applying robust regression, user feedback and global optimization techniques in an interactive loop visualized in Fig. 1. The loop constitutes a framework where we model the inherent uncertainty in the user feedback with Gaussian process regression and where we obtain user ratings through an intuitive and simple interface. Finally, we propose to use sequential optimization techniques to quickly find a (possibly local) optima of the preference function.

We evaluate the usefulness of the framework in a real-world experiment where personalization of a 5-band constant-Q equalizer (EQ) have been conducted for twenty-four test subjects. As the EQ has over fifty-nine thousands unique settings, the hypothesis is that the preferred settings will be hard to find without efficient sequential design. Furthermore, audio systems with a large number of parameters typically have an inherent correlation between particular parameters, for instance due to a finite number of frequency bands, for which correlation can be expected between adjacent frequency bands. In the particular EQ setting we may expect such correlation to exist between adjacent bands of the EQ and we therefore propose a specialized informative prior for including this knowledge into the system. The results from the real-world listening experiments focusing on the statistical difference between random experimentation and sequential experimental design, show a clear advantage of the sequential design approach.

The proposed framework is related to a number of studies in the field of preference learning and general personalization of audio systems. In particular Pardo *et. al.* [5] presents a similar system to ours, however the modeling framework presented in the present work is more flexible due to a non-parametric model, allows for easy inclusion of prior knowledge and provides a principled probabilistic way to make sequential decisions. Another related field is the individualization of hearing aids for which Birlutiu *et. al.* [2] propose a similar Gaussian process approach as ours, and furthermore consider the general notion of active learning. The main difference is the experimental paradigm which in [2] is based on pairwise comparisons. Furthermore, they do not evaluate the system in a real interactive loop and the active learning criterion is designed for generalization, not optimization as in our case.

Our contribution is thus two fold: First in Sec. 2, we propose a general personalization framework with an intuitive user interface (Sec. 2.3), a principled modeling approach using warped Gaussian processes (Sec. 2.1) and a sequential design method (Sec. 2.2). Secondly in Sec. 3, we evaluate the framework by an extensive listening experiment in a real world interactive setting and analyze the results. The results are discussed in Sec. 4 and the paper is concluded in Sec. 5.

2. FRAMEWORK

The proposed personalization approach uses an interactive loop to discover the user's preferred setting of a particular audio device, where we as an example use a 5-band constant-Q equalizer. The interactive loop is visualized in Fig. 1. The loop can conceptually be divided into three parts: a preference modeling part, a sequential design part and an interface part. The preference modeling part covers how to learn a preference function over equalizer settings based on user ratings. The sequential design part covers how to choose new equalizer settings based on what the model currently predicts. Finally, the interface part covers the design of the graphical user interface, such that it is both intuitive and easy to use for the users. The three parts are described in the following three sections.

2.1. Preference Model

We assume that for each system setting can be represented as a vector of parameters, $\mathbf{x}_i$ - and that for each unique setting we have a latent function value, $f(\mathbf{x}_i)$, expressing the user's preference for the particular setting. This function is to be learned trough a number of experiments where we observe the users expressed preference on a bounded scale, $y \in [0; 1]$. At some point we have evaluated n such distinct system settings $\mathbf{x}_i \in \mathbf{X}$ collected in

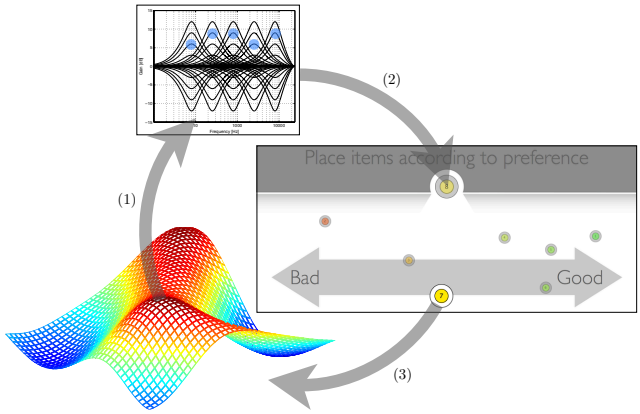


Fig. 1: Shows a conceptual overview of the interactive preference modeling system. At step (1) we draw a new equalizer from the current estimate of the subjects preference function. Next, at step (2) this particular equalizer is associated with a *ball*, in this case number *eight*, in the visualized user interface. Finally, after the user has rated the new equalizer, the preference function is updated to reflect current positions of all previous *balls*, this update occurs at step (3). We emphasize that the user at any time may select between previously sampled equalizers by clicking the *balls*, making the current song play through the newly selected equalizer.

$\mathbf{X} = \{\mathbf{x}_i | i = 1, ..., n\}$, and a related set of n responses denoted $\mathbf{Y} = \{y_i | i = 1, ..., n\}$.

We model the function which maps from settings, $\mathbf{x}_i$, to ratings, y_i, by a so-called warped Gaussian process [7]. A standard Gaussian process (GP) is a stochastic process defined as a collection of random variables, any finite subset of which must have a joint Gaussian distribution [6]. In effect, the GP is placed as a prior over any finite set of functional values $\mathbf{f} = [f_1, f_2, ..., f_n]^T$, where $f_i = f(\mathbf{x}_i)$, resulting in a finite multivariate Gaussian distribution over the set as $p(\mathbf{f}|\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where each element of the covariance matrix $\mathbf{K}$ is given by a covariance function $k(\cdot, \cdot)$ such that $[\mathbf{K}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. The GP prior can be used in non-parametric Bayesian regression frameworks where either the outputs or a likelihood can be parameterized by a smooth and continuous function $f(\cdot)$.

However, our regression setup is special due to the bounded nature of the ratings. We therefore use the warped Gaussian process in which the original data $\mathbf{Y}$ is transformed into a form where the data is modeled by a traditional Gaussian noise model [6]. Several warping functions would apply, but a natural choice is the inverse cumulative Gaussian (probit) $\Phi^{-1}(\cdot)$—with zero mean and unity variance—such that observation is warped as $z_i = \Phi^{-1}(y_i)$.

The final model is defined by,

$$\begin{aligned} \sigma_s|\theta_s &\sim \text{half student - t} \\ \sigma_\ell|\theta_\ell &\sim \text{half student - t} \\ f_i|\sigma_s, \sigma_\ell &\sim \mathcal{GP}\left(m\left(\mathbf{x}_i\right), \mathrm{k}\left(\mathbf{x}_i, \cdot\right)_{\sigma_s, \sigma_\ell}\right) \\ z_i|f_i &\sim \mathcal{N}\left(f_i, \sigma_i\right) \quad (1) \\ z_i &= \Phi^{-1}\left(y_i\right), \quad (2) \end{aligned}$$

where σ_ℓ is the length scale of the covariance function and σ_s is the variance of the latent function. We have placed hyper priors over the covariance parameters in order to provide a robust inference and

prediction scheme, especially in the sequential setup with relatively few observations. These hyper priors are half student-t distributions with parameters, $\theta = \{\xi, v\}$, with degree of freedom, ξ, and scale, v. We note that the observation noise, σ_i, can be included in the covariance function.

Given this model, the main questions remains regarding the covariance function, which effectively defines the smoothness of the function. We consider two covariance functions based on the general form of the squared exponential kernel

$$\mathrm{k}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_s \exp\left(-\frac{1}{\sigma_\ell}(\mathbf{x}-\mathbf{x}')^\top \mathbf{\Lambda}^{-1}\left(\mathbf{x}-\mathbf{x}'\right)\right) \tag{3}$$

In the first case, $\mathbf{\Lambda}$ is the identity matrix leading to the well-knoen (isotropic) squared exponential covariance function $\mathrm{k}_{\mathrm{iso}}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_s \exp\left(-\frac{1}{\sigma_\ell}\|\mathbf{x}-\mathbf{x}'\|^2\right)$. In the second case, $\mathbf{\Lambda}$ is a general positive semi-definite matrix defining a correlation between settings in input space as explicit prior information. We will denote this variant as the Mahanalobis covariance function, $\mathrm{k}_{\mathrm{mah}}\left(\mathbf{x}, \mathbf{x}'\right)$. The effect of the two options on the equalization example will be evaluated with reference to the standard case as `iso` and the Mahanalobis case as `mah`.

We turn to a standard GP inference scheme [6] in which the covariance parameters, σ_s, σ_ℓ, are approximated by point estimates by optimizing the marginal likelihood (or evidence) using a BFGS method and the posterior $p(\mathbf{f}|\mathbf{Y}, \mathbf{X})$ is analytical tractable. Extra terms are added to the standard evidence scheme [6] due to the student-t hyper priors. The predictive mean and (co)variance of the latent function, $\mathbb{E}(\mathbf{f}^*)$ and $\mathbb{V}(\mathbf{f}^*)$, are given in standard form as

$$\mathbb{E}\left\{\mathbf{f}^*\right\} = \mathbf{K}_{\mathbf{X}\mathbf{X}*}\left[\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma_i^2\mathbf{I}\right]^{-1}\Phi^{-1}\left(\mathbf{Y}\right) \tag{4}$$

$$\mathbb{V}\left\{\mathbf{f}^*\right\} = \mathbf{K}_{\mathbf{X}*\mathbf{X}*} - \mathbf{K}_{\mathbf{X}\mathbf{X}*}\left[\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma_i^2\mathbf{I}\right]^{-1}\mathbf{K}_{\mathbf{X}\mathbf{X}*} \tag{5}$$

where $\mathbf{K}_{\mathbf{A}\mathbf{B}}$ is the kernel matrix containing evaluations between training inputs, $\mathbf{A} = \mathbf{B} = \mathbf{X}$, test inputs, $\mathbf{A} = \mathbf{B} = \mathbf{X}*$, or between training and test inputs, $\mathbf{A} = \mathbf{X}*, \mathbf{B} = \mathbf{X}$.

The predictive distribution and in particular the predictive uncertainty is a clear advantage of the probabilistic GP framework, since the predictive mean and predictive variance can be used to determine the information gain in including a new candidate point into the mode, as considered next.

2.2. Sequential Experimental Design

Classic experimental design such as Latin Square or random experimentation [4] become increasing infeasible in high dimensions. As an alternative, we propose to use sequential design methods which, by greedy selection of the most informative next sample, potentially achieve much faster convergence than fixed designs [3].

The main purpose is to define a selection criterion which finds the optimal of the (unknown) preference function. The applied criterion is a slightly modified version of the so-called *Expected Improvement* (EI) [3], a known criterion in the design of computer experiment (DACE) community. The expected improvement is for each candidate point, $\mathbf{x}_j$, defined as,

$$EI(\mathbf{x}_j) = \sigma_{EI} \cdot \mathcal{N}\left(\frac{\mu_{EI}}{\sigma_{EI}}\right) + \mu_{EI} \cdot \Phi\left(\frac{\mu_{EI}}{\sigma_{EI}}\right), \tag{6}$$

where $\mathcal{N}(\cdot)$ is the standard Normal distribution and $\Phi(\cdot)$ is the standard cumulative Gaussian as before. Given the predictive distribution we can define,

$$\mu_{EI} = \mu_j - \mu_{\max}$$

$$\sigma_{EI}^2 = \sigma_j^2 + \sigma_{\max}^2 - 2\sigma_{j,\max}$$

where μ_j and σ_j is the predictive mean and variance of the test point and μ_m and σ_m is the predictive mean and variance of the current maximum of the preference function (using the predictive mean as the predictor), i.e., the current best setting, all of which originate from Eq. 4-5. The correlation between the two function values, $\sigma_{j,\max}$, requires correlated predictions which we refrain from due to computation burden, thus $\sigma_{j,\max} = 0, \forall \mathbf{x}_j$. Hence, the selection of a new point to evaluate is given by

$$\mathbf{x}_{new} = \arg\max_{\mathbf{x}_j} EI\left(\mathbf{x}_j\right)$$

which is then included in the current set of training points and evaluated by the user through the user

interface. We refer to this as the `active` configuration. A random configuration `rnd` is included in which samples are selected randomly to provide a baseline method.

The modeling framework leaves four strategies to be investigated experimentally: `rnd-iso`, `rnd-mah`, `active-iso` and `active-mah`.

2.3. Interface

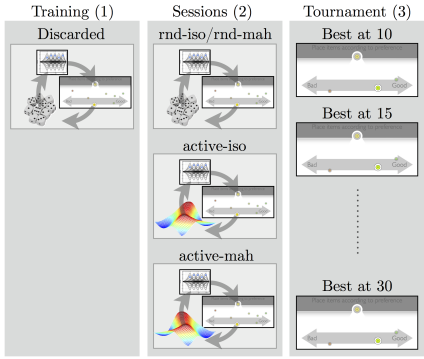
When applying absolute ratings, it is important to define anchor and/or reference points [1]. This allows subjects to compare stimuli with a fixed reference, such that each rating is *calibrated* both with respect to previous ratings, but also with respect to yet unobserved stimulus, which might redefine the end points of the rating scale. To address these two issues a graphical user interface similar to [5] is designed. Subjects can listen to previous equalizers (references) and are allowed to change previous ratings based on the new one. Obviously, this means that ratings are not **directly** comparable across subjects nor between iterations. However, it is not of particular interest to use ratings across subject to formulate one **single** optimal setting, but instead to we are interested in personalized solutions - one for each subject.

3. EXPERIMENT

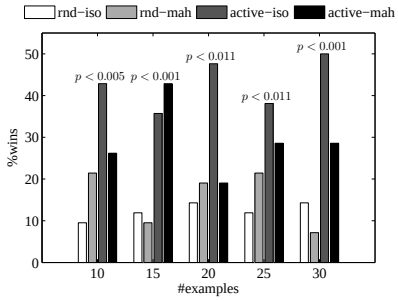
To evaluate the different model configurations and experimental designs in a real-world scenario, an experiment, in which the gains of a 5-band constant-Q equalizer is to be optimized by the 4 different versions of the proposed frameworks, was performed for 24 subjects. The procedure and results are described in the following sections.

3.1. Procedure

The experiment consisted of three parts: (1), (2) and (3) as visualized in Fig 2(a). During part (1), the user rates ten randomly chosen balls to learn how to use the interface and to get an impression of the stimuli (equalizer processed music). Part (2) consisted of three sessions for which the order of sessions was balanced across subjects. In each of the three sessions a particular model (`iso` or `mah`) and sequential design (`rnd` or `active`) is used to find a personalized setting of the EQ for the user. Finally in part (3), the preferred settings, found by each of the four combinations of models and sequential designs after 10, 15, 20, 25 and 30 presented



(a) Procedure



(b) Results

Fig. 2: (a): Visualization of the experiment with its 3 sessions: (1) Training, (2) Sessions and (3) Tournament. (b): The percentage of times the predicted preferred setting by each of the four models wins over the other models across users at each of the five tournament points.

settings, are determined by which model predicted the setting that is rated highest (in the tournament - see Fig. 2(a)). Each tournament (as defined in Fig. 2(a)) was repeated twice resulting in ten tournaments for which the sequence was randomized. In all parts, the sound was played back to the user through Sennheiser HD650 headphones and a Fire-

stoneAudio FUBAR DACIII headphone amplifier at constant level.

3.2. Results

The results are summarized in Fig. 2(b). The illustrated p-values gives the significance level for which the hypothesis, that the total number of active wins is equal to the total number of random wins at each tournament point (#examples), can be accepted.

4. DISCUSSION AND FUTURE WORK

Averaged across subjects and repetitions, active sequential design is significantly better than random design after any given number of examples, as shown by the p–values. This is without differentiating between the two applied covariance functions. It demonstrates the potential of the Bayesian model and active learning methods in audio applications. It is furthermore noted that a standard fixed design will approximate the random configuration in this high-dimensional space.

The second aspect is if the more informative *Mahalanobis* (`mah`) prior results in a more accurate model with few ratings available. This is generally not the case, however, after 15 examples the Mahalanobis prior has slightly more wins, which might indicate a small *window-of-opportunity* in which the more informative Mahalanobis prior actually prevails. It is speculated that in an even higher dimensional space ($\sim$10), this window will *grow*, rendering the *Mahalanobis* prior advantageous.

Future work will investigate the feasibility of inferring the correlation structure in the *Mahalanobis* prior. Finally, other response types, such as paired comparisons, is to be compared with the current absolute paradigm, which is also left for future work. We further plan to embed the current system in a slightly modified version as a Web application in the future.

5. CONCLUSION

We have proposed a method for obtaining true personalized systems—in particular audio systems—which utilizes the probabilistic modeling approach through sequential design. This improves the high-dimensional preference optimization procedure in respect to random (analogue to manual) experimentation. The solutions found by the sequential approach is significantly preferred over the solutions found by random experimentation. The results do not support any benefit in using the more informative Gaussian process prior with the Mahalanobis kernel compared to the less informative Gaussian process prior with the isotropic kernel.

6. REFERENCES

[1] Søren Bech and Nick Zacharov. *Perceptual Audio Evaluation - Theory, Method and Application*. Wiley, July 2006.

[2] A. Birlutiu, P. Groot, and T. Heskes. Efficiently learning the preferences of people. *Machine Learning*, (July 2010), May 2012.

[3] D. R Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[4] Douglas C. Montgomery. *Design and analysis of experiments*. Wiley, 2009.

[5] Bryan Pardo, David Little, and Darren Gergle. Building a personalized audio equalizer interface with transfer learning and active learning. *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies - MIRUM '12*, page 13, 2012.

[6] C.E. Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[7] E. Snelson, C. E. Rasmussen, and Z. Ghahramani. Warped gaussian processes. *Advances in Neural Information Processing Systems (NIPS)*, (16):337–344, 2004.

Bibliography

A. Anandkumar, D. Foster, D. Hsu, S. Kakade, and Y.-K. Liu. A spectral algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems 25*, pages 926–934, 2012.

R. Andersen and K. Lang. Communities from seed sets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 223–232, 2006.

R. Andersen, F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 475–486, 2006.

C. E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 1974. ISSN 00905364. doi: 10.2307/2958336.

G. H. Bakir, J. Weston, and B. Schölkopf. Learning to find pre-images. In *Advances in Neural Information Processing Systems 16*, pages 449–456. MIT Press, 2004.

M. J. Barber, M. Faria, L. Streit, and O. Strogan. Searching for communities in bipartite networks. AIP, 2008. doi: 10.1063/1.2956795.

C. F. Beckmann, M. DeLuca, J. T. Devlin, and S. M. Smith. Investigations into resting-state connectivity using independent component analysis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457): 1001–1013, 2005.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. Nvidia technical report, 2008.

P. Berkhin and J. D. Becher. Learning simple relations: Theory and applications. In *In Second SIAM Data Mining Conference*, 2002.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, October 2007. ISBN 0387310738.

M. Blaschko, J. Shelton, and A. Bartels. Augmenting feature-driven fmri analyses: Semi-supervised learning and resting state activity. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 126–134. 2009.

S. Bode and J.-D. Haynes. Decoding sequential stages of task preparation in the human brain. *NeuroImage*, 45(2):606–13, 2009. ISSN 1095-9572. doi: 10.1016/j.neuroimage.2008.11.031.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

V. D. Calhoun, T. Adali, G. D. Pearlson, and J. J. Pekar. A Method for Making Group Inferences from Functional MRI Data Using Independent Component Analysis. 151:140–151, 2001. doi: 10.1002/hbm.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

K. L. Clarkson, E. Hazan, and D. P. Woodruff. Sublinear optimization for machine learning. In *FOCS*, pages 449–457, 2010.

R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition in data: Diffusion maps. *Proc. Natl. Acad. Sci. USA*, 102(21):7426–7431, 2005.

S. Dambreville, Y. Rathi, and A. Tannenbaum. Statistical shape analysis using kernel PCA. In *IS&T/SPIE Symposium on Electrical Imaging*, 2006.

C. Davatzikos, K. Ruparel, Y. Fan, D. G. Shen, M. Acharyya, J. W. Loughead, R. C. Gur, and D. D. Langleben. Classifying spatial patterns of brain activity with machine learning methods: application to lie detection. *NeuroImage*, 28 (3):663–8, 2005. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2005.08.009.

I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. *KDD*, 2003.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 2001. ISBN 1-58113-391-X. doi: http://doi.acm.org/10.1145/502512.502550.

K. Duan, S. S. Keerthi, and A. N. Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41–59, 2003.

E. Eger, J. Ashburner, J. Haynes, R. Dolan, and G. Rees. fMRI activity patterns in human LOC carry information about object exemplars within category. *Journal of Cognitive Neuroscience*, 20(2):356–370, February 2008. ISSN 0898-929X. doi: 10.1162/jocn.2008.20019.

S. B. Eickhoff, K. E. Stephan, H. Mohlberg, C. Grefkes, G. R. Fink, K. Amunts, and K. Zilles. A new SPM toolbox for combining probabilistic cytoarchitectonic maps and functional imaging data. *NeuroImage*, 25(4):1325–35, May 2005. doi: 10.1016/j.neuroimage.2004.12.034.

A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, 2007.

S. Fortunato. Community detection in graphs. *Physics Reports*, 2010. ISSN 0370-1573. doi: DOI:10.1016/j.physrep.2009.11.002.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, 2004.

A. Frank and A. Asuncion. UCI machine learning repository, 2010.

K. J. Friston. The disconnection hypothesis. *Schizophrenia research*, 30(2): 115–25, Mar. 1998. ISSN 0920-9964.

K. J. Friston, a. P. Holmes, K. J. Worsley, J.-P. Poline, C. D. Frith, and R. S. J. Frackowiak. Statistical parametric maps in functional imaging: A general linear approach. *Human Brain Mapping*, 2(4):189–210, 1994. ISSN 1065-9471. doi: 10.1002/hbm.460020402.

W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114/115:815–839, 1989a.

W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its applications*, 114:815–839, 1989b.

J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1): 103–112, 2005.

S. Geyer, A. Ledberg, A. Schleicher, S. Kinomura, T. Schormann, U. Bürgel, T. Klingberg, J. Larsson, K. Zilles, and P. E. Roland. Two different areas within the primary motor cortex of man. 1996.

A. Gittens and M. W. Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *arXiv preprint arXiv:1303.1849*, 2013.

G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHUP, 2012.

T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for locally-biased learning. In *Advances in Neural Information Processing Systems 25*, pages 2537–2545, 2012.

T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for large-scale locally-biased learning. In *Submitted to Journal of Machine Learning Research*, 2013.

T. J. Hansen, T. J. Abrahamsen, and L. K. Hansen. A randomized heuristic for kernel parameter selection with large-scale multi-class data. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011a.

T. J. Hansen, M. Morup, and L. K. Hansen. Non-parametric co-clustering of large scale sparse bipartite networks on the gpu. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011b.

T. J. Hansen, L. K. Hansen, and K. H. Madsen. Decoding complex cognitive states online by manifold regularization in real-time fmri. In *Machine Learning and Interpretation in Neuroimaging*, pages 76–83. Springer, 2012.

T. J. Hansen, T. J. Abrahamsen, and L. K. Hansen. information-based kernel pca denoising by semi-supervised manifold learning. In *Submitted to Pattern Recognition Letters*, 2013.

J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 1972. ISSN 01621459. doi: 10.2307/2284710.

J.-D. Haynes, K. Sakai, G. Rees, S. Gilbert, C. Frith, and R. E. Passingham. Reading hidden intentions in the human brain. *Current biology : CB*, 17(4): 323–8, 2007. ISSN 0960-9822. doi: 10.1016/j.cub.2006.11.072.

H. Hindi. A Tutorial on Convex Optimization II: Duality and Interior Point Methods. *2006 American Control Conference*, (1):686–696, 2006. doi: 10.1109/ACC.2006.1655436.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *The Journal of educational psychology*, pages 498–520, 1933.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *JASA*, 2001.

T. Joachims. *The Maximum-Margin Approach to Learning Text Classifiers:Methods, theory, and algorithms*. PhD thesis, Dortmund University, 2001.

T. Joachims. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, volume 20, 2003.

Y. Kamitani and F. Tong. Decoding the visual and subjective contents of the human brain. *Nature neuroscience*, 8(5):679–85, 2005. ISSN 1097-6256. doi: 10.1038/nn1444.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, 2006.

G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, pages 495–502, 1970.

N. Kriegeskorte, R. Goebel, and P. Bandettini. Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the United States of America*, 103(10):3863–8, March 2006. ISSN 0027-8424. doi: 10.1073/pnas.0600244103.

S. Kumar, M. Mohri, and A. Talwalkar. Ensemble nystrom method. In *Neural Information Processing Systems*, volume 7, page 223. Citeseer, 2009.

J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. In *Proceedings of the International Conference on Machine Learning*, pages 408–415, 2003.

S. M. LaConte, S. J. Peltier, and X. P. Hu. Real-time fMRI using brain-state classification. *Hum Brain Mapp*, 28:1033–1044, October 2007. doi: 10.1002/hbm.20326.

P.-J. Lahaye, J.-B. Poline, G. Flandin, S. Dodel, and L. Garnero. Functional connectivity: studying nonlinear, delayed interactions between BOLD signals. *NeuroImage*, 20(2):962–74, Oct. 2003. ISSN 1053-8119. doi: 10.1016/S1053-8119(03)00340-9.

J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 695–704, 2008.

C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. WWW, 2010. ISBN 978-1-60558-799-8. doi: http://doi.acm.org/10.1145/1772690.1772760.

A. C. Lorena and A. C. P. L. F. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomput.*, 71:3326–3334, October 2008.

T. E. Lund, M. D. Nø rgaard, E. Rostrup, J. B. Rowe, and O. B. Paulson. Motion or activity: their role in intra- and inter-subject variation in fMRI. *NeuroImage*, 26(3):960–4, July 2005. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2005.02.021.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2004. ISSN 1545-5963. doi: 10.1109/TCBB.2004.2.

M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13:2339–2365, 2012.

S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2057–2064, 2011.

J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. ISSN 01628828. doi: 10.1109/34.868688.

P. Mannfolk, R. Wirestam, M. Nilsson, F. Stå hlberg, and J. Olsrud. Dimensionality reduction of fMRI time series data using locally linear embedding. *Magma (New York, N.Y.)*, 23(5-6):327–38, Dec. 2010. ISSN 0968-5243. doi: 10.1007/s10334-010-0204-0.

M. J. McKeown, S. Makeig, G. G. Brown, T.-P. Jung, S. S. Kindermann, A. J. Bell, and T. J. Sejnowski. Analysis of fmri data by blind separation into independent spatial components. Technical report, DTIC Document, 1997.

I. V. Mechelen, H. H. Bock, and P. D. Boeck. Two-mode clustering methods: a structured overview. *Statistical methods in medical research*, Oct. 2004.

S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems II*, pages 536–542. MIT Press, 1999.

J. Moeller and S. Strother. A regional covariance approach to the analysis of functional patterns in positron emission tomographic data. *Journal of Cerebral Blood Flow & Metabolism*, 11:A121–A135, 1991.

P. Morosan, J. Rademacher, A. Schleicher, K. Amunts, T. Schormann, and K. Zilles. Human primary auditory cortex: cytoarchitectonic subdivisions and mapping into a spatial reference system. *NeuroImage*, 13(4):684–701, 2001.

K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on*, 12(2):181–201, 2001.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS '01: Proceedings of the 15th Annual Conference on Advances in Neural Information Processing Systems*, 2001.

H. Nguyen. *GPU Gems 3*. 2007. ISBN 9780321545428.

J. B. Nielsen, B. S. Jensen, and T. J. Hansen. Fast and flexible elicitation of preference in complex audio systems. In *Work in progress; will be submitted to Audio Engineering Society*, 2013.

K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10 (9):424–430, 2006a.

K. a. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in cognitive sciences*, 10 (9):424–30, 2006b. ISSN 1364-6613. doi: 10.1016/j.tics.2006.07.005.

NVIDIA. *NVIDIA CUDA Programming Guide 2.0*. 2008.

E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

S. Ogawa, D. W. Tank, R. Menon, J. M. Ellermann, S.-G. Kim, H. Merkle, and K. Ugurbil. Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. *Proceedings of the National Academy of Sciences*, 89(13):5951–5955, 1992.

S. Papadimitriou and J. Sun. Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *ICDM*, 2008. ISBN 978-0-7695-3502-9. doi: http://dx.doi.org/10.1109/ICDM.2008.142.

J. Pitman. Combinatorial stochastic processes. 2002.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208: 1–21, 1999.

A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3): 430–452, 1990.

V. Ramanathan, W. Ma, V. T. Ravi, T. Liu, and G. Agrawal. Parallelizing an information theoretic co-clustering algorithm using a cloud middleware. *DMW*, 2010. doi: http://doi.ieeecomputersociety.org/10.1109/ICDMW.2010.100.

J. Reichardt and S. Bornholdt. Clustering of sparse data via network communities - a prototype study of a large online market. *Journal of Statistical Mechanics*, 2007. ISSN 1742-5468. doi: 10.1088/1742-5468/2007/06/P06016.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transcations of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 824–831. ACM, 2005.

A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. *Handbook of Comp. Molecular Biology*, 2004.

M. van den Heuvel, R. Mandl, and H. Hulshoff Pol. Normalized cut group clustering of resting-state FMRI data. *PloS one*, 3(4):e2001, Jan. 2008. ISSN 1932-6203. doi: 10.1371/journal.pone.0002001.

M. P. van den Heuvel and H. E. Hulshoff Pol. Exploring the brain network: a review on resting-state fMRI functional connectivity. *European*

neuropsychopharmacology : the journal of the European College of Neuropsychopharmacology, 20(8):519–34, Aug. 2010. ISSN 1873-7862. doi: 10.1016/j.euroneuro.2010.03.008.

C. Van Heerden and E. Barnard. Towards understanding the influence of svm hyperparameters. In *21st Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, pages 69–74, 2010.

V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Comput.*, 12:2013–2036, September 2000. ISSN 0899-7667. doi: 10.1162/089976600300015042.

M. Varewyck and J.-P. Martens. A Practical Approach to Model Selection for Support Vector Machines With a Gaussian Kernel. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(2):330–340, Apr. 2011. ISSN 1083-4419.

A. Venkataraman, K. R. Van Dijk, R. L. Buckner, and P. Golland. Exploring functional connectivity in fmri via clustering. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 441–444. IEEE, 2009.

R. L. Villars, C. W. Olofson, and M. Eastwood. Big data: What it is and why you should care. 2011.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17 (4):395–416, 2007.

G. Wahba. *Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV*, pages 69–88. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3.

C. Walder, R. Henao, M. Mørup, and L. K. Hansen. Semi-supervised kernel pca. *CoRR*, abs/1008.1398, 2010a.

C. Walder, R. Henao, M. Mørup, and L. K. Hansen. Semi-supervised kernel pca. *arXiv preprint arXiv:1008.1398*, 2010b.

H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *In SIGMOD*, 2002.

D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

C. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166, 2000.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Annual Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.

K.-P. Wu and S.-D. Wang. Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, 42:710–717, May 2009. ISSN 0031-3203.

S. Xiaoshan, J. Xiaoyu, H. Chongzhao, and L. Jianhua. Inter-class distance based kernel parameter evaluating method for rbf-svm. *Digital Manufacturing and Automation, International Conference on*, 1:853–858, 2010.

X. Xie, Z. Cao, and X. Weng. Spatiotemporal nonlinearity in resting-state fMRI of the human brain. *NeuroImage*, 40(4):1672–85, May 2008. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2008.01.007.

Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Learning infinite hidden relational models. *Uncertainity in Artificial Intelligence (UAI2006)*, 2006.

S. Y. K. Y. H.-P. K. Z. Xu, V.Tresp. Fast inference in inifinite hidden relational models. *Mining and Learning with Graphs (MLG'07)*, 2007.

X. Zhu. Semi-supervised learning literature survey. 2005.